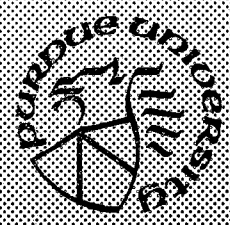
12-1-1989

On the Existence and Design of the Best Stack Filter Based Associative Memory

Pao-Ta Yu
Purdue University

Edward J. Coyle
Purdue University



On the Existence and Design of the Best Stack Filter Based Associative Memory

Pao-Ta Yu
Edward J. Coyle

TR-EE 89-72
December 1989

School of Electrical Engineering
Purdue University
West Lafayette, Indiana 47907

ON THE EXISTENCE AND DESIGN OF THE BEST STACK FILTER BASED ASSOCIATIVE MEMORY

Pao-Ta Yu

Edward J. Coyle

School of Electrical Engineering

Purdue University

West Lafayette, Indiana 47907

December 1989

ON THE EXISTENCE AND DESIGN OF THE BEST STACK FILTER BASED ASSOCIATIVE MEMORY

Pao-Ta Yu and Edward J. Coyle
School of Electrical Engineering
Purdue University
West Lafayette, IN 47907
(317) 494-3470

Abstract

The associative memory of a stack filter is defined to be the set of root signals of that filter. If the root sets of two stack filters both contain a desired set of patterns, but one filter's root set is smaller than the other, then the filter with the smaller root set is said to be *better* for that set of patterns. Any filter which has the smallest number of roots containing the specified set of patterns is said to be a *best* filter.

The configuration of the family of best filters is described via a graphical approach which specifies an upper and lower bound for the subset of possible best filters which are furthest from the sets of type-1 and type-2 stack filters. Knowledge of this configuration leads to an algorithm which can produce a near-best filter. This new method of constructing associative memories does not require the desired set of patterns to be independent and it can construct a much better filter than the methods in [1].

This work was supported by the National Science Foundation under the grant EET 87-21333.

1. Introduction

The root set of a stack filter is defined to be the set of all signals which are invariant under filtering by that stack filter [1-7]. In this paper, it will be considered to be the associative memory of that stack filter. This analogy with associative memories has led to a new notion of when one stack filter is *better* than another one [1]. Specifically, if the root sets of two stack filters both contain a desired set of patterns, but one filter's root set is smaller than the other, then this filter is said to be *better* for that set of patterns. The filter with the smaller set of roots would make the better associative memory.

As an example, consider the set of rank-order filters for some fixed window width. The root set of the median filter is the set of all signals in which constant-valued regions of some minimum length alternate with monotonic regions [5,6]. The root set of all other rank-order filters is the set of constant-valued signals [7], which is obviously a proper subset of the set of median filter roots. Therefore, if our goal is to find a filter which preserves the set of constant-valued signals, any rank order filter would work, but any rank-order filter other than the median would be better than the median filter. In other words, if the root set of the median filter is considered to be the set of memories of the median filter, then it contains many false memories [8], while the root sets of the other rank order filters contain no false memories.

This connection between associative memories and median/stack filters goes beyond the existence of invariant signals. It is also built on the convergence properties of median, rank-order, and stack filters. The median filter, rank order filters, and stack filters of type-1 and type-2 all filter every finite length input signal to a root signal after a finite number of iterations [1,2,5,7]. Such convergence behavior is an important feature of associative memories.

This paper strengthens this connection between stack filtering and associative memories by addressing the design of stack filter based associative memories which have the fewest false memories. We do not investigate the convergence properties of the filters which result from this procedure; that is still a problem requiring more research.

The input to our stack filter based associative memory will be the set of one-dimensional signals of length, L. The neural system to be considered will then consist of single layer of L neurons. Each neuron will be modeled as a stack filter with $2N+1$ inputs and a single output. The $2N+1$ inputs will be $2N+1$ consecutive points in the input signal. We will also require each of these L stack filters to be the same; i.e., all the neurons in our single layer perform the same operation. In this way, the output of our single layer of L neurons

can be considered to be the output obtained by passing a stack filter of window width $2N+1$ over the length L input signal by the standard method of advancing the window by one position along the signal at each time increment.

For simplicity, only the case of binary input signals will be presented; the case of multi-valued signals can be obtained by exploiting the weak superposition property known as the threshold decomposition property of median filters, rank order filters, and stack filters [9,10].

Since each stack filter is defined by a positive Boolean function, the use of binary signals means that each "neuron" in our model is actually a positive Boolean function. Note that some positive Boolean functions are threshold logic gates, but that cascades of threshold logic gates may be required to implement the remaining positive Boolean functions. In summary, in this model a positive Boolean function will be treated as a cascade of nonlinear thresholding operator, which is the definition of a neuron in the conventional neural network models.

The primary goal of this paper is then to develop algorithms for designing stack filters which have root sets which contain the smallest possible superset of a prespecified set of patterns. Since only binary inputs will be considered, this problem reduces to finding the positive Boolean function which preserves a specified set of binary inputs. The restriction to positivity, which is necessary and sufficient for these filters to have the superposition property mentioned above, plays a key role in the analysis and in the design algorithms.

Some initial results on this problem of designing stack filter based associative memories were obtained in [1]. It presented a classification scheme for stack filters, some basic tools for the characterization of the root-preservation behavior of the different classes of stack filters, and procedures for finding best type-1 and type-2 stack filters.

In this paper, we extend those results in two ways. We obtain a characterization of those filters which are best amongst all stack filters with respect to preserving a desired set of patterns; and a heuristic learning scheme is proposed which produces a stack filter which is better than any of the filters proposed in [1]. This learning scheme, which is a consequence of the characterization of the set of best filters, produces a type-3 stack filter which is better than the best type-1 and type-2 stack filters produced by the learning algorithms in [1].

The characterization of the set of possible best filters is obtained in the form of a Hasse diagram. Each filter preserving the desired set of patterns

belongs to a family of better filters satisfying some appropriate condition. Each node on the Hasse diagram that will be constructed represents a best filter chosen from each family. This diagram describes a few of the possible best filters and then tight bounds which are guaranteed to enclose a nonempty subset of the set of best stack filters. Significantly, this subset of best stack filters is in a certain sense the best filters which are furthest from the sets of type-1 and type-2 stack filters.

As mentioned earlier, one issue that is not addressed in this paper is the convergence behavior of the filters produced by the algorithms developed in this paper. Since type-3 stack filters are being considered, instead of type-1 or type-2 filters as in [1], we are not guaranteed that every input will be filtered to a root signal; oscillations may occur. Also, the algorithms currently do not provide a mechanism for specifying to which root a given nonroot signal will be filtered. The primary contribution of this paper is to see, once a window width has been specified, how well the best stack filter of that window width can do in terms of the number of false memories.

This paper is organized as follows. In the next section, we review some basic concepts and collect all the operators used in [1,2] to form an algebra, called the *stacking algebra*. Some new properties of this stacking algebra are also introduced in this section.

In Section 3, we review the definition of a better filter and extend the results shown in [1]. Two significant better filters are introduced and the prototype of two procedures used to construct these filters are emphasized.

In Section 4, we design two procedures extended from Section 3 to produce better filters which are better than the filters mentioned in Section 3. Then, the configuration of the subset of best filters which are furthest from the type-1 and type-2 stack filters is described. The theorem presenting this characterization is called the *Existence Theorem*, and it is important since it completely specifies the associative memory capability of stack filters.

In Section 5, an *Invariance Theorem* is presented. It specifies the behavior of all the best filters characterized in the Existence Theorem.

In Section 6, we use the Existence and Invariance Theorems to design a heuristic learning scheme which can construct much better filters than the filters mentioned in Section 3. Finally, some conclusions are given.

2. Basic Concepts

In this section, the definition of binary stack filters is reviewed and the relationship between associative memories and a stack filters is made precise. Second, the classification of stack filters into four different types is presented. It allows us to specify the behavior of stack filters which preserve a desired set of patterns. Third, a useful set algebra, called the *stacking algebra*, is extended from [1,2]. This algebra is the primary tool used in this paper. Finally, the *root–preservation lemmas* first developed in [1,2] are reviewed and simplified. A more detailed discussion of some of the material in this section is found in [1,2]. This section does contain material not found in [1,2].

2.1. Binary Stack Filters

For simplicity, all signals we will deal with in this paper will be binary signals. Once the behavior of a stack filter is understood for binary signals, its behavior for multi-valued signals can be obtained through the weak superposition property called the threshold decomposition [9,10]. We thus concentrate on binary stack filters, which are just stack filters with binary inputs.

A signal X of length L will be denoted by the vector $X = (x_1, x_2, \ldots, x_L)$, where $x_i \in \{ 0, 1 \}$; $i=1,2,\ldots,L$. The appended version of X for some positive integer N is called X' and is defined as follows:

$$X' = (x'_{-N+1}, x'_{-N+2}, \cdots, x'_{L+N})$$

in which

$$x'_i = x_1, \quad -N+1 \leq i \leq 0;$$

$$x'_i = x_i, \quad 1 \leq i \leq L;$$

$$x'_i = x_L, \quad L+1 \leq i \leq L+N.$$

Clearly, $X' \in \{ 0, 1, \}^{L+2N}$ and is just X with its end points repeated N times. In the sequel, X' will be called the appended version of X.

The definition of a stack filter $S_f(\cdot)$ is based on a positive Boolean function f of $2N+1$ variables [1-4] which can be expressed as a Boolean expression containing no complements of the input variables [11]. The exact definition of stack filters is stated in the following.

The input signal X corresponding to the output signal Y is defined as follows:

$$y_i = f(x'_{i-N}, x'_{i-N+1}, \ldots, x'_i, \ldots, x'_{i+N}) \qquad i=1,\ldots,L. \tag{2.1}$$

Note that x'_i is the ith component of the appended version of X and y_i is the ith component of the output signal Y obtained from the stack filter based on the window function f.

From Eq.(2.1) it is clear that this sliding window filter can be represented by a mapping of the form $S_f(X) = Y$; that is, $S_f:\{0, 1\}^L \rightarrow \{0, 1\}^L$.

As stated above, a stack filter is based on a positive Boolean function. Any Boolean function can be completely specified by two subsets, its *on−set* and its *off−set*, which are the set of binary strings of length $2N+1$ for which the filter's output is a one or a zero, respectively. More precisely, if f is a Boolean function of n variables, the *on−set* and *off−set* of f will be denoted as $on(f)$ and $off(f)$, respectively, and are defined as follows:

$$on(f) = \{ v : v \in \{0, 1\}^n, f(v)=1 \}$$

and

$$off(f) = \{ v : v \in \{0, 1\}^n, f(v)=0 \}$$

There is a trivial partial ordering of stack filters formed by their *on−sets* (or *off−sets*) under the relationship of set inclusion.

Definition 2.1:

Let f and g be two Boolean functions of n variables. "$f \leq g$" if and only if $on(f) \subseteq on(g)$; equivalently, "$f \leq g$" if and only if $off(g) \subseteq off(f)$.

□

Clearly, the relation "$\leq$" defined in Definition 2.1 forms a partial ordering. If f and g are positive, we also can say "$S_f(\cdot) \leq S_g(\cdot)$" if and only if $f \leq g$.

Note that the notation $\leq$ is the conventional notation to represent a partially ordered relation no matter on which set we define. For instant, the partial ordering defined between two length L vectors X and Y is denoted as $X \leq Y$ if $x_i \leq y_i$; $i=1,2,\ldots,L$.

For convenience, the partial ordering $\leq$ defined on the set A is denoted as $(A, \leq)$ and is called a partially ordered set.

Definition 2.2:

Let $(A, \leq)$ be a partially ordered set. An element $m \in A$ is minimal (maximal) when there is no other element a in A for which $a \leq m$ ($m \leq a$). If the minimal (maximal) element is unique, then it is called the least

(greatest) element of $(A, \leq)$.

□

In the sequel, we assume the length of signals is fixed, say L, and the window width of any stack filter that is being considered is $2N+1$; that is, the number of variables of the positive Boolean functions is $2N+1$.

The set of fixed points of the stack filter $S_f(\cdot)$ is called its root set and is denoted as $R(f)$; that is,

$$R(f)=\{X:X\in\{0,1\}^L \text{ and } S_f(X)=X\}.$$

The root set of a stack filter will be considered to be an associative memory which is based on that stack filter.

2.2. The Classification of Stack Filters [1]

Four different types of stack filters are identified in this subsection. They are called type-0 through type-3 and are defined in terms of the characteristics of the *on−sets* of the filters in that type.

The set diagrams, shown in Figure 2.1, are used to clarify their characteristics. A circle is used to represent the whole space $\{0,1\}^{2N+1}$ or x^{2N+1}, the upper semicircle is used to represent the subspace x^N1x^N which is the set consisting of all vectors whose central component is "1", and the lower semicircle is used to represent the subspace x^N0x^N which is the set consisting of all vectors whose central component is "0". Note that the highest and the lowest points in this diagram are 1^{2N+1} and 0^{2N+1}, respectively. The shaded area represents the possible case of *on−set* of f on which the stack filter $S_f(\cdot)$ is based. The more detailed description of x^N1x^N and x^N0x^N will be given in the next subsection.

If the *on−set* of f is the empty set, x^N1x^N, or x^{2N+1}, the stack filter $S_f(\cdot)$ is called a type-0 stack filter. The set diagrams of *on−set* on which type-0 stack filters are based are shown in Figure 2.1 (a). The class of type-0 stack filters is denoted as TP_0.

If the *on−set* of f is a proper subset of x^N1x^N, then the stack filter $S_f(\cdot)$ is called a type-1 stack filter. A set diagram of the *on−set* of a typical type-1 stack filter is shown in Figure 2.1 (b). The class of type-1 stack filters is denoted as TP_1.

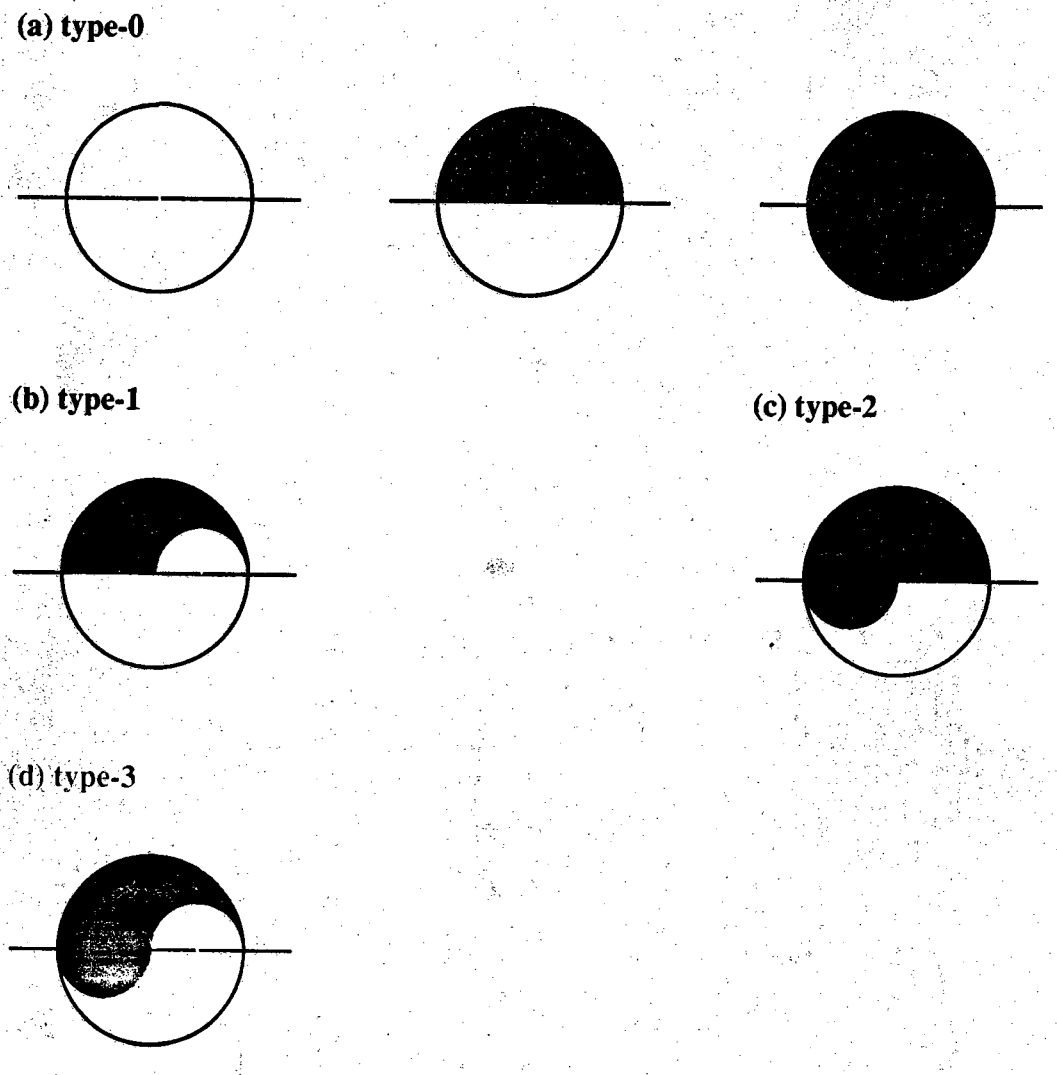


Figure 2.1: Each circle represents the space x^{2N+1}. The upper half of the circle is the subspacex^N1x^N; the lower half is x^N0x^N. The shaded area in each circle represents the on-set of a Boolean function of $2N+1$ variables. (a) The three type-0 stack filters arebased on positive Boolean functions whose on-set are empty set, x^N1x^N, x^{2N+1}. (b) The on-sets of the Boolean functions which represent type-1 stack filters are a proper subset of x^N1x^N. (c)The off-sets of the Boolean functions which represent type-2 stack filters are a proper subset of x^N0x^N.(d) Type-3 stack filters are based on positive Boolean functions whose on-sets lies across the boundary between x^N1x^N and x^N0x^N.

If the *on−set* of f is a proper superset of $x^N 1 x^N$, then the stack filter $S_f(\cdot)$ is called a type-2 stack filter. A set diagram of the *on−set* of a typical type-2 stack filter is shown in Figure 2.1 (c). The class of type-2 stack filters is denoted as TP_2.

If the stack filters are not type-0 , type-1, or type-2, then they are classified as type-3 stack filters. A set diagram of the *on−set* of a typical type-3 stack filter is shown in Figure 2.1 (d). The class of type-3 stack filters is denoted as TP_3.

2.3. Stacking Algebra and Basic Properties

A stacking algebra, $(\sim\!\mathcal{P}, P\_EXT, N\_EXT, D^0, D^1)$, consists of a set $\sim\!\mathcal{P}$, and four operators $P\_EXT(\cdot)$, $N\_EXT(\cdot)$, $D^0(\cdot)$, and $D^1(\cdot)$. We first introduce a set notation for vectors in the space $\{0,1\}^n$, and then define the stacking algebra and present some its properties.

2.3.1. Cubic Expressions

Cubic expressions can be used to explain Boolean functions in terms of geometry or sets in n-dimensional space. As will be seen later, this geometrical interpretation is particularly helpful in the study of stack filters.

The Cartesian product of n copies of $\{0,1\}$, $\{0,1\}^n = \{0,1\}\times\{0,1\}\times\cdots\times\{0,1\}$, is called the *n−cube*, where n is a positive integer. It is obviously a subset of the Euclidean n-space R^n. There are 2^n elements in the *n−cube*, and each one is called a vertex.

These 2^n vertices are the 2^n different valuations of the ordered n-string $x_1 x_2 \cdots x_n$, where $x_i \in \{0,1\}$ for all i from 1 to n. Thus, vertices and vectors (or signals) represent the same thing, and, therefore, in $\{0,1\}^n$, the noun vertex is synonymous with vector.

With the notion of vertices and *n−cube*s, subcubes of the *n−cube*s can be defined. First, a vertex will be called a *0−cube* since it is a single point and therefore dimensionless. A *1−cube* is a subset of the *n−cube* obtained by replacing any one component of any vertex by "x", where "x" means "0" or "1". For example, $0x11 = \{\ 0011,\ 0111\ \}$ is a *1−cube* and can be thought of as a line between two vertices. A *2−cube* is a subset of *n−cube* replacing any two components of any vertex by x's. For example, $0x1x = \{\ 0010,\ 0011,\ 0110,$

0111 } is a 2−*cube*, and can be thought of as a square. Continuing on, an r−*cube* is a subset of the n−*cube* obtained by replacing r components of any vertex by x's.

Note that an r−*cube* is a subset of the n−*cube* except when $r=0$, and that the number of elements in an r−*cube* is 2^r. For convenience, we will often treat the 0−*cube* as a singleton subset of the n−*cube*.

2.3.2. Stacking Algebra

Now, we define the stacking algebra $(\frown\!\!\mathcal{P}, P\_EXT, N\_EXT, D^0, D^1)$ as follows.

$\frown\!\!\mathcal{P}$ is the set of all subsets of $\{0,1\}^{2N+1}$; that is, $\frown\!\!\mathcal{P}$ is the power set of $\{0,1\}^{2N+1}$. Note that $2N+1$ is the window width of stack filter.

Definition 2.3:

Let $P\_EXT(\cdot)$ be a function mapping from $\frown\!\!\mathcal{P}$ to $\frown\!\!\mathcal{P}$. It satisfies the following conditions:

(a) $P\_EXT(\varnothing) = \varnothing$, where $\varnothing$ is the empty set.

(b) If v is vertex in $\{0,1\}^{2N+1}$, then $P\_EXT(\{v\})$ is the r−*cube* obtained by replacing all the 0 components of v by x's, where r is the number of 0's in v.

(c) If $V \in \frown\!\!\mathcal{P}$ and V is nonempty, then

$$P\_EXT(V) = \bigcup_{v \in V} P\_EXT(\{v\}).$$

□

Definition 2.4:

Let $N\_EXT(\cdot)$ be a function mapping from $\frown\!\!\mathcal{P}$ to $\frown\!\!\mathcal{P}$. It satisfies the following conditions:

(a) $N\_EXT(\varnothing) = \varnothing$, where $\varnothing$ is the empty set.

(b) If v is vertex in $\{0,1\}^{2N+1}$, then $N\_EXT(\{v\})$ is the r−*cube* obtained by replacing all the 1 components of v by x's, where r is the number of 1's in v.

(c) If $V \in \frown\!\!\mathcal{P}$ and V is nonempty, then

$$N\_EXT(V) = \bigcup_{v \in V} N\_EXT(\{v\}).$$

□

Although these operators are set operators, we will, for convenience, use the notation $P\_EXT(v)$ and $N\_EXT(v)$ instead of $P\_EXT(\{v\})$ and $N\_EXT(\{v\})$ when the argument v is a vertex.

Example 2.1:

$P\_EXT(\ 111\ ) = \{\ 111\ \}$. $P\_EXT(\ 110\ ) = 11x = \{\ 110, 111\ \} = 1^2x$. $P\_EXT(\ 000\ ) = xxx = x^3 = \{0,1\}^3$. Note from the preceding expressions that if there are several consecutive 0's, 1's, or x's in a cubic expression, then we would like to write them in a exponential form. $N\_EXT(\ 111\ ) = xxx = x^3 = \{0,1\}^3$. $N\_EXT(\ 110\ ) = xx0 = x^20$. $N\_EXT(\ 000\ ) = \{\ 000\ \}$.

$$P\_EXT(\{\ 101, 100\ \}) = P\_EXT(\ 101\ ) \cup P\_EXT(\ 100\ )$$

$$= 1x1 \cup 1xx$$

$$= 1xx$$

$$N\_EXT(\{\ 101, 100\ \}) = N\_EXT(\ 101\ ) \cup N\_EXT(\ 100\ )$$

$$= x0x \cup x00$$

$$= x0x$$

□

Definition 2.5:

Let $C \in \widetilde{\mathcal{P}}$. Then $D^0(\cdot)$ and $D^1(\cdot)$ are defined as follows:
$D^0(C) = \{u0v\colon \forall\ u1v \in C\ \}$ and $D^1(C) = \{u1v\colon \forall\ u0v \in C\ \}$.

□

The operator $D^0(\cdot)$ is used to mirror a vertex in the subspace x^N1x^N to the subspace x^N0x^N and the operator $D^1(\cdot)$ is used to mirror a vertex in the subspace x^N0x^N to the subspace x^N1x^N.

Example 2.2:

Let $C=\{000,011,101,110\}$ in space $\{0,1\}^3$, then $D^0(C)=\{001,100\}$ such that 011 and 110 are mirrored to 001 and 100, respectively, and $D^1(C)=\{010,111\}$ such that 000 and 101 are mirrored to 010 and 111, respectively.

□

Extremal Property: [1,2]

(a) Let $V\in\widetilde{\mathcal{P}}$. $w\in P\_EXT(V)$ if and only if there exists a $v\in V$ such that $v\leq w$.

(b) Let $V\in\widetilde{\mathcal{P}}$. $w\in N\_EXT(V)$ if and only if there exists a $v\in V$ such that $w\leq v$.

□

The upward cone is used to represent the set diagram of $P\_EXT(v)$ with the tip v at the bottom, and the downward cone is used to represent the set diagram of $N\_EXT(v)$ with tip v at the top. Both are shown in Figures 2.2 (a) and (b). From the graphical representation, it is obvious that the vertex w is above the vertex v -- any v with $v\leq w$ -- will be covered by the upward cone formed by $P\_EXT(v)$. Similarly, any vertex w below the vertex v -- any w with $w\leq v$ -- will be covered by the downward cone formed by $N\_EXT(v)$.

Note that these graphical representations are essential tools for the work in this paper; they also make the results much easier to understand. Their use can be seen in the following result on disjoint subsets.

Disjoint Property:

(a) Let $V\in\widetilde{\mathcal{P}}$ and $W\subseteq[\{0,1\}^{2N+1}-P\_EXT(V)]$. Then $N\_EXT(W)\cap P\_EXT(V)=\emptyset$.

(b) Let $V\in\widetilde{\mathcal{P}}$ and $W\subseteq[\{0,1\}^{2N+1}-N\_EXT(V)]$. Then $P\_EXT(W)\cap N\_EXT(V)=\emptyset$.

(c) Let $V, W\in\widetilde{\mathcal{P}}$. If $V\cap W=\emptyset$, then $D^0(V)\cap D^0(W)=\emptyset$ and $D^1(V)\cap D^1(W)=\emptyset$.

Proof:

(a) If $N\_EXT(W)\cap P\_EXT(V))\neq\emptyset$, then there exists a $v\in N\_EXT(W)\cap P\_EXT(V)$
$\Longrightarrow$ there exists $w\in W$ such that $v\leq w$

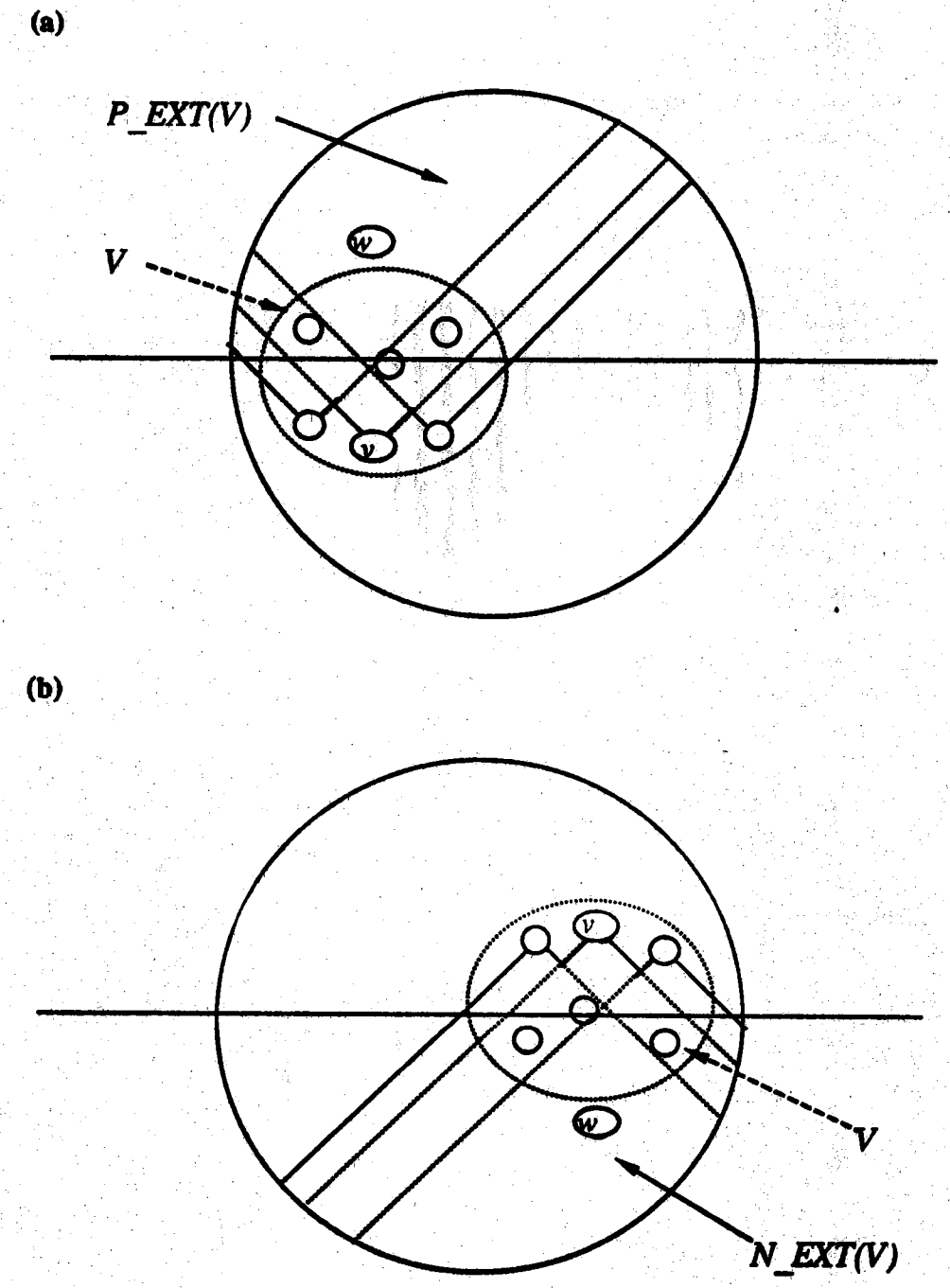


Figure 2.2: The graphical representation of a set V after taking positive and negative extensions is shown in above (a) and (b), respectively. From those graphs the phenomena of extreme points according to the effect of both extensions are also specified by a special extreme point v and an other point w covered by $P\_EXT(v)$ or $N\_EXT(v)$.

$\Rightarrow w \in P\_EXT(V)$ by the Extremal Property
$\Rightarrow w \in W \cap P\_EXT(V)$.
This contradicts $P\_EXT(V) \cap W = \varnothing$.
Therefore, $N\_EXT(W) \cap P\_EXT(V) = \varnothing$.

(b) Same arguments as in (a).

(c) This is a trivial consequence of Definition 2.5.

□

A graphical illustration of the Disjoint Property is shown in Figure 2.3.

Positivity Property:[1,2]

The following three conditions are equivalent:

(a) f is a positive Boolean function of $2N+1$ variables.

(b) $P\_EXT(on(f)) = on(f)$.

(c) $N\_EXT(off(f)) = off(f)$.

□

This property also reveals an important result: $P\_EXT^2(on(f)) = P\_EXT(on(f))$ and $N\_EXT^2(off(f)) = N\_EXT(off(f))$ even when f is not a positive Boolean function. Therefore, any Boolean function can be processed by taking $P\_EXT(\cdot)$ or $N\_EXT(\cdot)$ to be a positive Boolean function. This is why we call the preceding property the Positivity Property.

Based on the Positivity Property, applying $P\_EXT(\cdot)$ or $N\_EXT(\cdot)$ to any V in $\sim\mathcal{P}$ yields a nice structure, which will be called Positive-Structure or P-Structure, which is distributed over the subspaces $x^N 1 x^N$ and $x^N 0 x^N$ in the fashion shown in Figures 2.2 and 2.3.

P-Structure Property:

Let $V \in \sim\mathcal{P}$. Then the subsets $P\_EXT(V)$ and $N\_EXT(V)$ of $\{0,1\}^{2N+1}$ will have the following properties:

(a) $D^1(P\_EXT(V) \cap x^N 0 x^N) \subseteq [P\_EXT(V) \cap x^N 1 x^N]$ or
$D^0([x^N 1 x^N - P\_EXT(V)]) \subseteq [x^N 0 x^N - P\_EXT(V)]$.

(b) $D^0(N\_EXT(V) \cap x^N 1 x^N) \subseteq [N\_EXT(V) \cap x^N 0 x^N]$ or
$D^1([x^N 0 x^N - N\_EXT(V)]) \subseteq [x^N 1 x^N - N\_EXT(V)]$.

Proof:

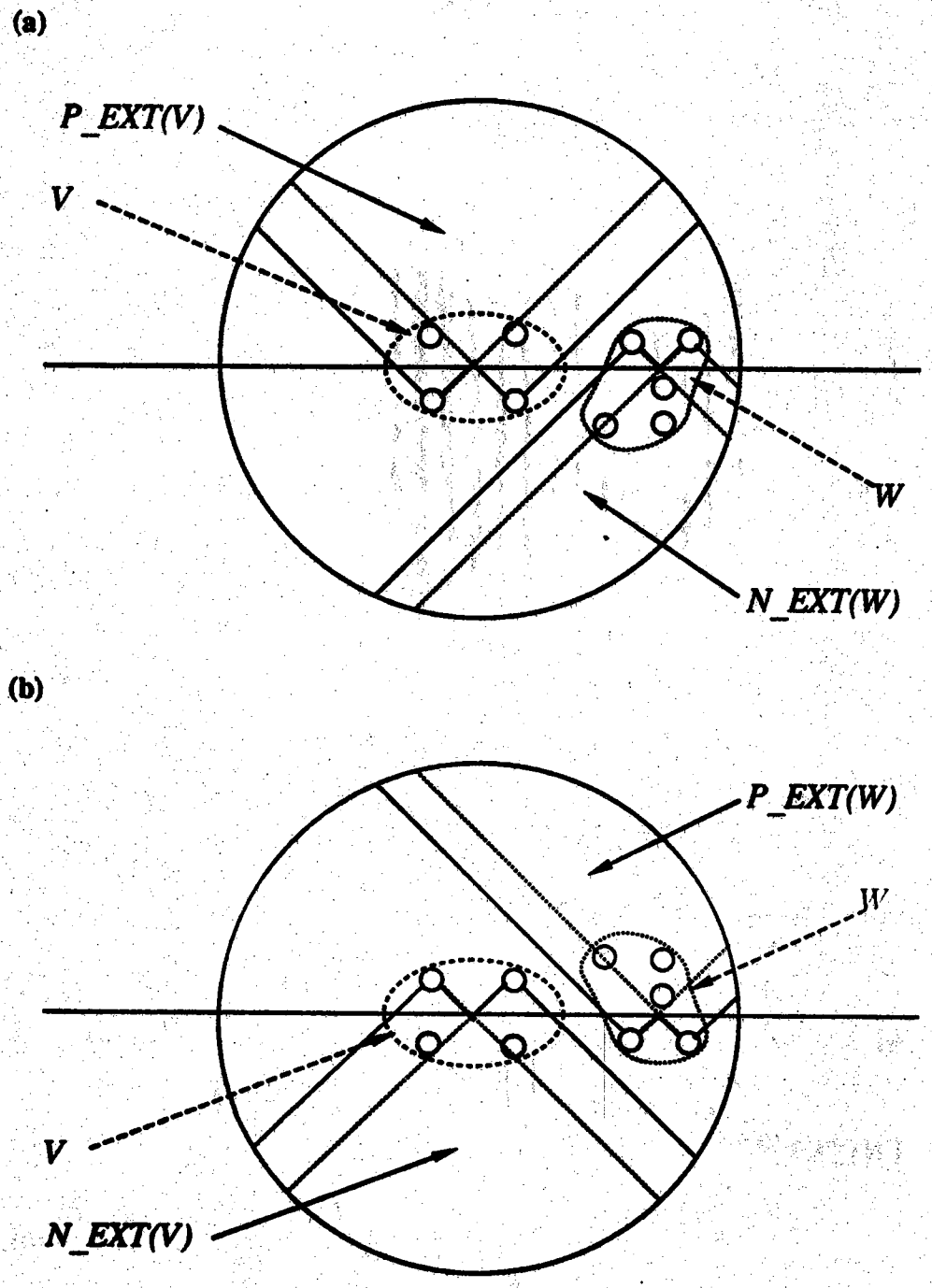


Figure 2.3: Figures (a) and (b) illustrate the phenomenon of Disjoint Property. (a) The intersection of *P_EXT(V)* and *N_EXT(W)* is empty. (b) The intersection of *N_EXT(V)* and *P_EXT(W)* is empty.

(a) For all $u1v \in D^1(P\_EXT(V) \cap x^N 0 x^N)$, we have $u0v \in [P\_EXT(V) \cap x^N 0 x^N]$. Then $u1v \in P\_EXT(V)$ because of the relation $u0v \leq u1v$ and the Extremal Property. Since the center of $u1v$ has the value 1, it is trivially in $x^N 1 x^N$. Thus, $u1v \in [P\_EXT(V) \cap x^N 1 x^N]$.
Similarly, it can be shown that $D^0([x^N 1 x^N - P\_EXT(V)]) \subseteq [x^N 0 x^N - P\_EXT(V)]$.

(b) This follows from arguments similar to those in (a).

□

2.4. Root-Preservation Lemma and Some Properties

In this subsection, the binary version of the root-preservation lemma in [1] stated concisely. Only this concise statement will be needed to derive the later results in this paper.

Definition 2.6:

The *one−set* obtained with a window of width $2N+1$ from a specific binary signal X is the set of subsignals of length $2N+1$ which are contained in the appended version of X and whose center point has value "1". We denote it as $one(X;2N+1)$. The *zero−set* obtained with a window of width $2N+1$ from a specific binary signal X is the set of subsignals of length $2N+1$ which are contained in the appended version of X and whose center point has value "0". We denote it as $zero(X;2N+1)$.

□

Note that X is a finite length signal -- say of length L -- and assume there is a window of width $2N+1$ sliding from left to right on the appended version of X. If the center point in the window has value 1, then this window vector belongs to $one(X;2N+1)$; otherwise, it belongs to $zero(X;2N+1)$. We represent these sets more precisely as follows:

$$one(X;2N+1) = \{ (x'_{i-N}, x'_{i-N+1}, \ldots, x'_i, \ldots, x'_{i+N}) : x'_i = 1,\ i \in \{1,2,\ldots,L\} \}$$

and

$$zero(X;2N+1) = \{ (x'_{i-N}, x'_{i-N+1}, \ldots, x'_i, \ldots, x'_{i+N}) : x'_i = 0,\ i \in \{1,2,\ldots,L\} \}.$$

For simplicity, we write $one(X)$ and $zero(X)$ instead of $one(X;2N+1)$ and $zero(X;2N+1)$, since $2N+1$ is the default window width.

Example 2.3:

Let $X = 11001001$ and consider a window of width 3. The appended signal corresponding to X is 1110010011. Note that for each end there is a "1" appended.

$$one\text{-}set\ of\ X\ with\ window\ of\ width\ 3$$

$$= one(X)$$

$$= \{ 111, 110, 010, 011 \}$$

$$zero\text{-}set\ of\ X\ with\ window\ width\ 3$$

$$= zero(X)$$

$$= \{ 100, 001, 100, 001 \}$$

$$= \{ 100, 001 \}$$

□

Recall that if a stack filter $S_f(\cdot)$ satisfies $S_f(X)=X$, then we say that $S_f(\cdot)$ preserves X, or X is a root (fixed point) of $S_f(\cdot)$.

Root-Preservation Lemma:[1]

A stack filter $S_f(\cdot)$ based on the positive Boolean function f of $2N+1$ variables preserves a set of specific binary signals A if and only if $P\_EXT(\bigcup_{X\in A} one(X)) \subseteq on(f)$ and $N\_EXT(\bigcup_{X\in A} zero(X)) \subseteq off(f)$.

□

For convenience, we let $one(A)=\bigcup_{X\in A} one(X)$ and $zero(A)=\bigcup_{X\in A} zero(X)$.

Some properties based on the classification of stack filters are summarized below. Those properties can be directly shown by applying the root-preservation lemma.

Property 2.1:[1]

Let $S_f(\cdot)$ and $S_g(\cdot)$ be two type-1 stack filters. If $on(f) \subseteq on(g)$, then $R(f) \subseteq R(g)$.

□

Property 2.2:[1]

Let $S_f(\cdot)$ and $S_g(\cdot)$ be two type-2 stack filters. If $off(f) \subseteq off(g)$, then $R(f) \subseteq R(g)$.

□

Property 2.3:[1]

Let $S_f(\cdot)$ and $S_g(\cdot)$ be two type-3 stack filters. If $(on(f) \cap x^N 1 x^N) \subseteq (on(g) \cap x^N 1 x^N)$ and $(off(f) \cap x^N 0 x^N) \subseteq (off(g) \cap x^N 0 x^N)$, then $R(f) \subseteq R(g)$

□

The relationship between f and g discussed in Property 2.3 can be seen in Figure 2.4. The *on−set* of f in the subspace $x^N 1 x^N$ is a subset of the *on−set* of g in the subspace $x^N 1 x^N$, but the *on−set* of f in the subspace $x^N 0 x^N$ is a superset of the *on−set* of g in the subspace $x^N 0 x^N$.

Any type-3 stack filter $S_f(\cdot)$ can be decomposed into a type-1 stack filter $S_g(\cdot)$ and a type-2 stack filter $S_h(\cdot)$ as follows:

$$f(x_1, x_2, \ldots, x_{2N+1}) = g(x_1, x_2, \ldots, x_{2N+1}) + \bar{x}_{N+1} h(x_1, x_2, \ldots, x_{2N+1})$$

where $on(g) = (on(f) \cap x^N 1 x^N)$ and $on(h) = (on(f) \cap x^N 0 x^N) \cup x^N 1 x^N$. The set diagram of this decomposition is shown in Figure 2.5.

Now, we state a property with regard to this decomposition as follows.

Property 2.4: [1]

$R(f) \subseteq R(g)$ and $R(f) \subseteq R(h)$.

□

3. Better Filters

If the goal is to design a stack filter for the purpose of recognizing a set of patterns, the design procedure should produce a stack filter whose root set is as close as possible to the set of patterns to be recognized. It would, in fact, be desirable if the root set contains every pattern to be recognized. This can almost never be accomplished with a stack filter without the root set being a strict superset of the set of desired pattern. In other words, there will be some spurious roots, which can be called "false memories".

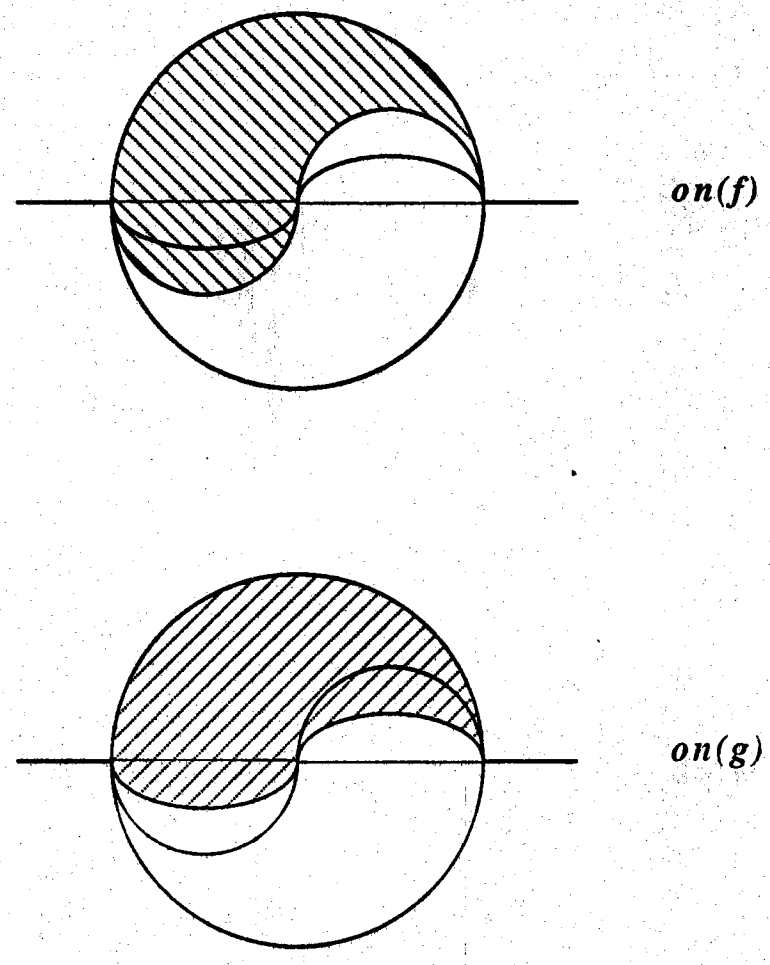


Figure 2.4: The relationship between *on(f)* and *on(g)* is shown above. The portion of *on(f)* in the subspace x^N1x^N is a subset of the portion of *on(g)* in x^N1x^N. The portion of *on(g)* in the subspace x^N0x^N is a subset of the portion of *on(f)* in x^N0x^N.

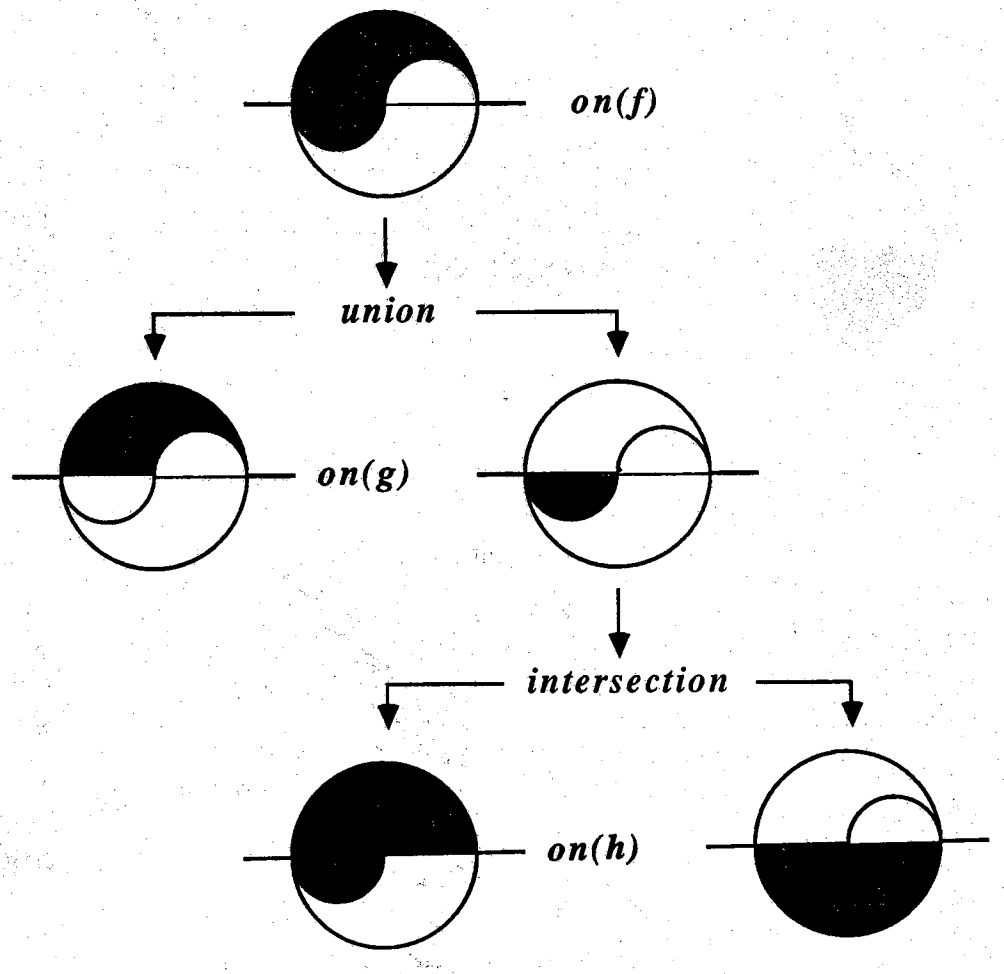


Figure 2.5: The decomposition of *on(f)* into *on(g)* and *on(h)*. $on(f)=on(g)\cup(on(h)\cap x^N 0 x^N)$.

Although these spurious roots can not be avoided entirely, their number can be minimized. As in [1], a filter with the fewest number of spurious roots will be called a *best* stack filter. A filter which has fewer spurious roots than another filter is said to be *better* than that other filter. Note that there may be more than one *best* filter; two *best* filters will have the same number of roots, and both will have root sets containing the desired set of patterns, but their roots sets will not perfectly coincide.

The concept of a *better filter* was first proposed in [1] and some properties of these filters were stated in that paper. In this paper, we will extend the results in [1] by specifying the configuration of the family of possible best filters, by presenting techniques for eliminating false memories, and by proposing a heuristic algorithm to find a near-best stack filter.

Definition 3.1:

Suppose that stack filters $S_f(\cdot)$ and $S_g(\cdot)$ both preserve a desired set of patterns, A . We say "$S_f(\cdot)$ is better than $S_g(\cdot)$ with respect to A" if $|R(f)| \leq |R(g)|$ where $|\cdot|$ is the cardinality of a set. We denote "$S_f(\cdot)$ is better than $S_g(\cdot)$ with respect to A" as $f \leq_A g$. Note that A is a subset of $\{0, 1\}^L$.

□

Thus, a *best* stack filter with respect to A is one which has the smallest number of roots or memories which are not in A. That is, the root set of best filter is the closest to the desired set of patterns.

Note the use of the "$\leq$" in the definition of a "better" stack filter. The possibility of equality must be allowed if the concept of a *best* filter is to be tractable. In other words, there may be several filters which are *best*. This lack of uniqueness and the necessity of the equality is a consequence of the difference between the two "orderings" of the length $2N+1$ binary vectors that can be observed in the window of the filter. The first ordering is the partial ordering defined by the stacking property. The second is the ordering defined by the motion of the filter window -- if 11001 is in the window now, then when the window moves to the left by one position only two vectors could be in the window at its new position, either 10010 or 10011. Because of this difference, it is possible for two stack filters to have the same number of roots yet there may actually be more length $2N+1$ binary vectors for which the output of one of the filters agrees with the middle bit in the vector [12].

The relation "$\leq_A$" is not a partial ordering, because it does not satisfy the property of antisymmetry. For example, max and min filters in the class of rank-order filters preserve the constant roots [7] and we know $|R(\max)| = |R(\min)| = 2$; i.e., $\max \leq_{const} \min$ and $\min \leq_{const} \max$, but $\max \neq \min$.

In the remainder of this paper, we will investigate the family of stack filters which preserve a desired set of patterns, called set A.

Assumption:

Assume that the *one−set* of A does not contain the element 0^N10^N and also assume that the *zero−set* of A does not contain the element 1^N01^N.

□

Without this assumption, the only stack filter $S_f(\cdot)$ which can preserve a set A containing these elements is based on $f(x_1, x_2, \ldots, x_{2N+1}) = x_{N+1}$ [2], and is therefore a trivial filter (the identity filter). To avoid this situation, enlarge the window width $2N+1$, increase the density of encoding pixels L, or remove those signals from A which contain the pattern 0^N10^N or 1^N01^N.

In the sequel, we denote $ST(A)$ as the set of stack filters which preserve A; that is,

$$ST(A) = \{S_f(\cdot) : \forall X \in A,\ S_f(X) = X\}.$$

Property 3.1:[1]

$\forall S_g(\cdot) \in ST(A) \cap TP_1$, $f_{t1} \leq_A g$ where $on(f_{t1}) = P\_EXT(one(A))$.

□

That is, the stack filter $S_{f_{t1}}(\cdot)$, based on the positive Boolean function f_{t1} whose *on−set* is $P\_EXT(one(A))$, is a best filter among all the type-1 stack filters with respect to A.

Example 3.1:

Let set A be the set of monotonic signals of length 15; i.e., $L=15$, and $2N+1=5$. That is,

$$A = \{0^n1^{15-n} : n = 0, 1, \ldots, 15\} \cup \{1^n0^{15-n} : n = 0, 1, \ldots, 15\}.$$

$one(A) = \{11100, 11110, 11111, 01111, 00111\}$. Thus, $P\_EXT(one(A)) = 111xx \cup xx111$. Then the stack filter $S_{f_{t1}}(\cdot)$ based on $f_{t1}(x_1, x_2, x_3, x_4, x_5)$

$= x_1x_2x_3+x_3x_4x_5$ is best among all type-1 stack filters which preserve the monotonic signals -- no other type-1 filter can have fewer roots, although there may be some which have the same number of roots.

□

Property 3.2:[1]

$\forall S_g(\cdot)\in ST(A)\cap TP_2$, $f_{t2}\leq_A g$ where $off(f_{t2})=N\_EXT(zero(A))$.

□

That is, the stack filter $S_{f_{t2}}(\cdot)$, based on the positive Boolean function f_{t2} whose *off−set* is $N\_EXT(zero(A))$, is a best filter among all the type-2 stack filters with respect to A.

Example 3.2:

Let A, L, and $2N+1$ be same as shown in the preceding example. $zero(A)=\{11000,10000,00000,00001,00011\}$. Thus, $N\_EXT(zero(A)) = xx000 \cup 000xx$. Then the stack filter $S_{f_{t2}}(\cdot)$ based on $f_{t2}(x_1,x_2,x_3,x_4,x_5)$ $= \overline{x_3x_4x_5}+\overline{x_1x_2x_3}$ is a stack filter which is best among all type-2 stack filters which preserve the monotonic signals. Note that after it is simplified $f_{t2}(x_1,x_2,x_3,x_4,x_5) = x_3+x_1x_4+x_1x_5+x_2x_4+x_2x_5$.

□

Since we know that a stack filter is based on a positive Boolean function, we will often refer to a positive Boolean function as a stack filter. That is, in this paper, the terms "stack filter" and "positive Boolean function" mean the same thing. Since we are only considering binary signals in this paper, this terminology is consistent, and is convenient.

f_{t1} is a type-1 stack filter, which means that its *on−set* is completely within the subspace x^N1x^N is $P\_EXT(one(A))$; no portion of it lies in x^N0x^N. f_{t2} is a type-2 stack filter, which means that its *off−set* is completely within the subspace x^N0x^N is $N\_EXT(zero(A))$; no portion of it lies in x^N1x^N. In the following we will be interested in finding stack filters which preserve A when these constraints on the on- and off-sets are removed. In other words, we are interested in type-3 stack filters, in which case the *on−set* and *off−set* will contain elements from x^N0x^N and x^N1x^N, respectively (see Figure 2.1(d)).

The following lemma states some of the behavior encountered the on- and off-sets are extended to x^N0x^N or x^N1x^N, respectively.

Lemma 3.1:

(a) Let $V=D^0(P\_EXT(one(A))) \cap [x^N0x^N-N\_EXT(zero(A))]$ and assume it is nonempty. Then $P\_EXT(one(A)\cup V) = P\_EXT(one(A)) \cup V$; that is, the only points in x^N0x^N which are in $P\_EXT(one(A)\cup V)$ are those which are already in V. Also, trivially, if $W\subseteq V$, then $P\_EXT(one(A)\cup W)=P\_EXT(one(A))\cup P\_EXT(W) \subseteq P\_EXT(one(A)) \cup V$.

(b) Let $V=D^1(N\_EXT(zero(A))) \cap [x^N1x^N-P\_EXT(one(A))]$ and assume it is nonempty. Then $N\_EXT(zero(A)\cup V) = N\_EXT(zero(A)) \cup V$; that is, the only points in x^N1x^N which are in $N\_EXT(zero(A)\cup V)$ are those which are already in V. Also, trivially, if $W\subseteq V$, then $N\_EXT(zero(A)\cup W) =N\_EXT(zero(A))\cup N\_EXT(W) \subseteq N\_EXT(zero(A)) \cup V$.

Proof:

(a) $P\_EXT(one(A)\cup V) = P\_EXT(one(A)) \cup P\_EXT(V)$. Clearly, $[P\_EXT(one(A))\cup V] \subseteq [P\_EXT(one(A))\cup P\_EXT(V)]$.
Thus, we only need to show that $P\_EXT(one(A))\cup V \supseteq P\_EXT(one(A)) \cup P\_EXT(V)$; i.e., we need to show
$\forall\, v\in[P\_EXT(one(A))\cup P\_EXT(V)] \Longrightarrow v\in[P\_EXT(one(A))\cup V]$.
Case 1: If $v=p0q$ where p and $q \in \{0,1\}^N$, then $v\in P\_EXT(V)$ since the central component of every vertex in $P\_EXT(one(A))$ has the value "1". Therefore, there exists an element $r0s\in V$, where r and $s \in \{0,1\}^N$, such that $r0s\leq p0q$ (by the Extremal Property).
Since $r0s\in D^0(P\_EXT(one(A)))$, $r1s\in P\_EXT(one(A))$ and $r1s\leq p1q$. This implies that $p1q\in P\_EXT(one(A))$, which, in turn, implies $p0q\in D^0(P\_EXT(one(A)))$.
By the Disjoint Property, $p0q\notin N\_EXT(zero(A))$; that is, $p0q\in[x^N0x^N-N\_EXT(zero(A))]$.
This implies $v=p0q\in \{D^0(P\_EXT(one(A))) \cap [x^N0x^N-N\_EXT(zero(A))]\} =V$.
Therefore, $v\in[P\_EXT(one(A))\cup V]$.
Case 2: If $v=p1q\in[P\_EXT(one(A))\cup P\_EXT(V)]$, then $p1q\in P\_EXT(one(A))$ or $p1q\in P\_EXT(V)$.
If $p1q\in P\_EXT(one(A))$, then trivially $v=p1q\in[P\_EXT(one(A))\cup V]$.
If $p1q\in P\_EXT(V)$, then there exists $r0s\in V$ such that $r0s\leq p1q$. This implies that $r1s\in P\_EXT(one(A))$ and $r1s\leq p1q$, which, in turn,

implies that $v=p1q\in P\_EXT(one(A))$.
Therefore, $\forall\ v\in[P\_EXT(one(A))\cup P\_EXT(V)] \rightarrow v\in[P\_EXT(one(A))\cup V]$.

(b) The proof follows the same line of reasoning as in part (a).

□

In Figure 3.1(a), the area $\hat{c}$ represents $one(A)$ and the areas $\hat{b}$ and $\hat{c}$ represent $P\_EXT(one(A))$. Then the areas $\hat{h}$ and $\hat{g}$ represent $D^0(P\_EXT(one(A)))$. Significantly, area $\hat{h}$ represents $V=D^0(P\_EXT(one(A)))\cap[x^N0x^N-N\_EXT(zero(A))]$. Similarly, the meaning of each region in Figure 3.1(b) can refer to the description shown below Figure 3.1(b).

It would be nice to be able to precisely illustrate the Lemma 3.1 with the type of set diagrams used in Figures 3.1(a) and (b). This is not possible, though, since these figures are limited to two dimensions and the lemma describes a higher dimensional phenomenon. The best we can say with these diagrams is that part (a) of Lemma 3.1 states that $P\_EXT(\hat{h})$, the positive extension of the set $\hat{h}$, contains no elements of the sets $\hat{a}$ and $\hat{f}$ shown in Figure 3.1(a). The dual statement, from Figure 3.1(b), is that $N\_EXT(\hat{a})$, the negative extension of the set $\hat{a}$, contains no elements of the sets $\hat{c}$ and $\hat{h}$.

Lemma 3.1 will now be used to prove two properties which will be important for specifying the set of possible best filters. These properties will lead to two filters: one will form a lower bound on the subset of possible best filters which are furthest from the sets of type-1 and type-2 stack filters; the other will form the upper bound.

Property 3.3:

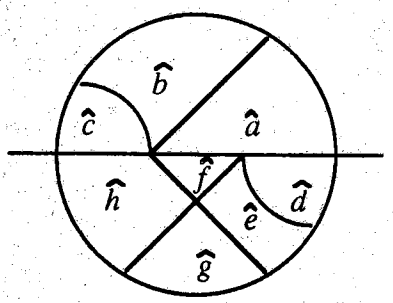
Let $V=D^0(P\_EXT(one(A)))\cap[x^N0x^N-N\_EXT(zero(A))]$, in which case V is given by the set $\hat{h}$ in Figure 3.1(a). Also let $on(f)=P\_EXT(one(A))\cup V$. Then,
$\forall S_g(\cdot)\in ST(A)\cap TP_1$, $f\leq_A g$. It means that f is better than any type-1 filter.

Proof: Appendix.

□

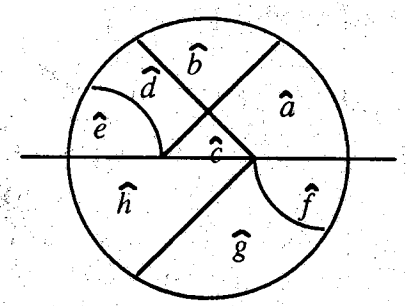
We denote this *better* stack filter f as f_l; it will be the lower bound we are interested in. This filter will be a type-3 stack filter if $V\neq\emptyset$; otherwise, $f_l=f_{t1}$.

(a)



$\hat{c}=one(A)$
$\hat{b}\cup\hat{c}=P\_EXT(one(A))$
$\hat{d}=zero(A)$
$\hat{d}\cup\hat{e}\cup\hat{g}=N\_EXT(zero(A))$
$\hat{f}\cup\hat{h}=x^N0x^N\text{-}(\hat{d}\cup\hat{e}\cup\hat{g})$
$\hat{g}\cup\hat{h}=D^0(P\_EXT(one(A)))$
$\hat{h}=(\hat{f}\cup\hat{h})\cap(\hat{g}\cup\hat{h})$

(b)



$\hat{e}=one(A)$
$\hat{b}\cup\hat{d}\cup\hat{e}=P\_EXT(one(A))$
$\hat{f}=zero(A)$
$\hat{f}\cup\hat{g}=N\_EXT(zero(A))$
$\hat{a}\cup\hat{c}=x^N1x^N\text{-}(\hat{b}\cup\hat{d}\cup\hat{e})$
$\hat{a}\cup\hat{b}=D^1(N\_EXT(zero(A)))$
$\hat{a}=(\hat{a}\cup\hat{b})\cap(\hat{a}\cup\hat{c})$

Figure 3.1: The meaning of each region in the above graphs is described below each graph.

Note that the *on−set* of f_l is given in Figure 3.1(a) by $\hat{b}\cup\hat{c}\cup\hat{h}$. The precise form of the *off−set* of f_l will be specified in Lemma 3.2 (a).

Example 3.3:

Let A, L, and $2N+1$ be same as shown in Example 3.1.
$D^0(P\_EXT(one(A)))=110xx\cup xx011=\hat{g}\cup\hat{h}$. $(x^20x^2-N\_EXT(zero(A)))=$ $\{11001,11010,\ 11011,\ 10001,10010,10011,\ 01001,01010,01011\}\ =\ \hat{f}\cup\hat{h}$. Then, $\hat{h}=$ $\{01011,10011,11001,11010,11011\}$. Thus, $on(f_l)=P\_EXT(one(A)\cup\hat{h})\ =P\_EXT(one(A))\cup\hat{h}\ =xx111\ \cup\ x1x11\ \cup$ $1xx11\ \cup\ 11xx1\ \cup\ 11x1x\ \cup\ 111xx$. That is, $S_{f_l}(\cdot)$ based on $f_l(x_1,x_2,x_3,x_4,x_5)\ =\ x_1x_2x_3\ +x_1x_2x_4\ +x_1x_2x_5\ +x_1x_4x_5\ +x_2x_4x_5$ $+x_3x_4x_5$ is better than any type-1 stack filters.

□

We have shown how to find a filter which is better than any type-1 filter, we now derive dual results which lead to a filter which is better than any type-2 filter.

Property 3.4:

Let $W=D^1(N\_EXT(zero(A)))\cap[x^N1x^N-P\_EXT(one(A))]$, in which case W is given by the set $\hat{a}$ in Figure 3.1(b). Also let $off(f)=N\_EXT(zero(A))\cup W$. Then,
$\forall S_g(\cdot)\in ST(A)\cap TP_2$, $f\leq_A g$. It means that f is better than any type-2 filter.

Proof: Appendix.

□

We denote this *better* stack filter f as f_u; it will be the upper bound we are interested in. This filter will be a type-3 stack filter if $W\neq\emptyset$; otherwise, $f_u=f_{t2}$. Note that the *off−set* of f_u is given in Figure 3.1(b) by $\hat{a}\cup\hat{f}\cup\hat{g}$. The precise form of the *on−set* of f_u will be specified in Lemma 3.2(b).

Example 3.4:

Let A, L, and $2N+1$ be same as shown in Example 3.1.
$D^1(N\_EXT(zero(A;5)))=xx100\cup001xx=\hat{a}\cup\hat{b}$. $(x^21x^2-P\_EXTone(A)))=$ $\{00100,\ 00101,00110,\ 01100,01101,01110,10100,10101,10110\}\ =\ \hat{a}\cup\hat{c}$. Then, $\hat{a}=$ $\{00100,00101,\ 00110,01100,10100\}$. Thus,

$off(f_u)=N\_EXT(zero(A)\cup\hat{a})$

$=N\_EXT(zero(A))\cup\hat{a}$ $=000xx\cup\ 00x0x\cup00xx0\ \cup0xx00\ \cup x0x00\cup xx000.$
That is, $S_{f_u}(\cdot)$ based on $f_u(x_1,x_2,x_3,x_4,x_5) = x_1x_4 + x_1x_5 + x_2x_4 + x_2x_5 + x_1x_2x_3 + x_3x_4x_5$ is better than any type-2 filters.

□

The intersection of the on-set of f_l with x^N1x^N is the same as the intersection of the on-set of f_{t1} with x^N1x^N. f_l is the *greatest* (in the sense of the partial ordering defined in Definition 2.2) filter satisfying this relationship with f_{t1}.

The intersection of the off-set of f_u with x^N0x^N is the same as the intersection of the off-set of f_{t2} with x^N0x^N. f_u is the *least* (in the sense of the partial ordering defined in Definition 2.2) filter satisfying this relationship with f_{t2}.

Lemma 3.2:

(a) $off(f_l)=N\_EXT([x^N1x^N-P\_EXT(one(A))]\cup zero(A))$ or $off(f_l)=N\_EXT(\hat{a}\cup\hat{d})$ where $\hat{a}$ and $\hat{d}$ are shown in Figure 3.1(a).

(b) $on(f_u)=P\_EXT(one(A)\cup[x^N0x^N-N\_EXT(zero(A))])$ or $on(f_u)=P\_EXT(\hat{e}\cup\hat{h})$ where $\hat{e}$ and $\hat{h}$ are shown in Figure 3.1(b).

(c) $on(f_l)\subseteq on(f_u)$; that is, $S_{f_l}(\cdot)\leq S_{f_u}(\cdot)$.

Proof:

(a) We will show that $P\_EXT(\hat{c}\cup\hat{h})\ \cup\ N\_EXT(\hat{a}\cup\hat{d})=\{0,1\}^{2N+1}$ and $P\_EXT(\hat{c}\cup\hat{h})\cap N\_EXT(\hat{a}\cup\hat{d})=\emptyset$. Then, $off(f_l)=N\_EXT(\hat{a}\cup\hat{d})$.
First, we show that $P\_EXT(\hat{c}\cup\hat{h})\cup N\_EXT(\hat{a}\cup\hat{d})=\{0,1\}^{2N+1}$.

$$P\_EXT(\hat{c}\cup\hat{h})\cup N\_EXT(\hat{a}\cup\hat{d})$$

$$=P\_EXT(\hat{c})\cup P\_EXT(\hat{h})\cup N\_EXT(\hat{a})\cup N\_EXT(\hat{d})$$

$$=(\hat{b}\cup\hat{c}\cup\hat{h})\cup(\hat{a}\cup N\_EXT(\hat{a}))\cup(\hat{d}\cup\hat{e}\cup\hat{g})$$

$$=(\hat{a}\cup\hat{b}\cup\hat{c}\cup\hat{d}\cup\hat{e}\cup\hat{g}\cup\hat{h})\cup N\_EXT(\hat{a}).$$

Note that $\hat{b}\cup\hat{c}\cup P\_EXT(\hat{h})=\hat{b}\cup\hat{c}\cup\hat{h}$ by Lemma 3.1.
If we can show that $\hat{f}\subseteq N\_EXT(\hat{a})$, then $P\_EXT(\hat{c}\cup\hat{h})\cup N\_EXT(\hat{a}\cup\hat{d})=\{0,1\}^{2N+1}$.
If $\hat{f}=\emptyset$, the result follows trivially, so assume $\hat{f}\neq\emptyset$.

By the specification of each area in Figure 3.1(a), $\hat{f}= x^N0x^N-N\_EXT(zero(A)) \quad -D^0(P\_EXT(one(A)))$. So, $\forall v\in\hat{f}$, $v\notin D^0(P\_EXT(one(A)))$.

Now, we want to show $\hat{f}\subseteq N\_EXT(\hat{a})$. If there exists a $v\in\hat{f}$ such that $v\notin N\_EXT(\hat{a})$, then $\forall y\in\hat{a}$, $v\not\leq y$ (recall the Extremal Properties we stated in Section 2).

Hence, $(P\_EXT(v)\cap x^N1x^N)\cap\hat{a}=\emptyset$; that is, all the vertices in the subspace x^N1x^N which are greater than or equal to v must not belong to $\hat{a}=x^N1x^N-P\_EXT(one(A))$. Therefore, $(P\_EXT(v)\cap x^N1x^N)\subseteq P\_EXT(one(A))$. By P-Structure Property, $(P\_EXT(v)\cap x^N0x^N)\subseteq D^0(P\_EXT(one(A)))$; that is, $v\in D^0(P\_EXT(one(A)))$. This contradicts $v\notin D^0(P\_EXT(one(A)))$. Therefore, $\hat{f}\subseteq N\_EXT(\hat{a})$. Hence, $P\_EXT(\hat{c}\cup\hat{h})\cup N\_EXT(\hat{a}\cup\hat{d}) = \{0,1\}^{2N+1}$.

Second, we show that $P\_EXT(\hat{c}\cup\hat{h})\cap N\_EXT(\hat{a}\cup\hat{d})=\emptyset$.

$$
\begin{aligned}
&P\_EXT(\hat{c}\cup\hat{h})\cap N\_EXT(\hat{a}\cup\hat{d})\\
&\quad=(\hat{b}\cup\hat{c}\cup\hat{h})\cap[N\_EXT(\hat{a})\cup(\hat{d}\cup\hat{e}\cup\hat{g})]\\
&\quad=[(\hat{b}\cup\hat{c}\cup\hat{h})\cap N\_EXT(\hat{a})]\cup[(\hat{b}\cup\hat{c}\cup\hat{h})\cap(\hat{d}\cup\hat{e}\cup\hat{g})]\\
&\quad=\{[(\hat{b}\cup\hat{c})\cup\hat{h}]\cap N\_EXT(\hat{a})\}\cup\emptyset\\
&\quad=[(\hat{b}\cup\hat{c})\cap N\_EXT(\hat{a})]\cup[\hat{h}\cap N\_EXT(\hat{a})].
\end{aligned}
$$

By Disjoint Property, $(\hat{b}\cup\hat{c})\cap N\_EXT(\hat{a})=\emptyset$.

We now claim that $\hat{h}\cap N\_EXT(\hat{a})=\emptyset$, and show this by using " proof by contradiction".

If $r0s\in\hat{h}\cap N\_EXT(\hat{a})$ where r and $s\in\{0,1\}^N$ then $r1s\in(\hat{b}\cup\hat{c})$ and there exists $t1u\in\hat{a}$ such that $r0s\leq t1u$ where r, s, t, and $u\in\{0,1\}^N$. This implies that $r1s\leq t1u$, which, in turn, implies that $r1s\in N\_EXT(\hat{a})\cap x^N1x^N=\hat{a}$. This then implies that $r1s\in\hat{a}\cap(\hat{b}\cup\hat{c})$. But, since $\hat{a}\cap(\hat{b}\cup\hat{c})=\emptyset$, this is a contradiction.

(b) The proof of this statement is the dual of the proof in (a).

(c) By Lemma 3.2 (a) and (b) and the diagrams shown in Figures 3.1 (a) and 3.1 (b), this result is obvious.

□

With this Lemma we have now exactly specifies the structure of $S_{f_l}(\cdot)$ and $S_{f_u}(\cdot)$; that is, we know $on(f_l)$ and $off(f_u)$ from Lemma 3.1, and Lemma 3.2 gives us $off(f_l)$ and $on(f_u)$.

Summary:

(a) $S_{f_l}(\cdot)$ is a best stack filter among the class of stack filters which preserve the set of desired patterns A and whose $on-set$ in the subspace $x^N 1 x^N$ is equal to $P\_EXT(one(A))$.

$$on(f_l)=P\_EXT(one(A))\cup \{D^0(P\_EXT(one(A)))\cap[x^N 0 x^N - N\_EXT(zero(A))]\}$$

$$off(f_l)=N\_EXT([x^N 1 x^N - P\_EXT(one(A))]\cup zero(A))$$

(b) $S_{f_u}(\cdot)$ is a best stack filter among the class of stack filters which preserve the set of desired patterns A and whose $off-set$ in the subspace $x^N 0 x^N$ is equal to $N\_EXT(zero(A))$.

$$on(f_u)=P\_EXT(one(A)\cup[x^N 0 x^N - N\_EXT(zero(A))])$$

$$off(f_u)=N\_EXT(zero(A)) \cup\{D^1(N\_EXT(zero(A)))\cap[x^N 1 x^N - P\_EXT(one(A))]\}.$$

In the next section we will produce better filters than f_l and f_u through a procedure which is a generalization of the procedures in this section to find f_l and f_u.

4. The Existence Theorem for Best Filters

Our goal is to find a stack filter which has the smallest number of root signals while still having a desired set of patterns as roots. One difficulty in determining such a filter is that there is no closed form expression for the number of root signals of a stack filter. There are some methods of counting the number of roots for the median filter for various window widths [13-15] and closed form expressions are available for certain special cases [13]. A general, but not computationally tractable, technique for counting the number of roots of many filters is given in [16]. Still, a general and tractable technique is not available for this problem.

In this section, we find filters which tightly bound a nonempty subset of the set of possible best stack filters. If we had a tractable method of counting the number of roots of a stack filter we could go further and possibly determine the *entire* set of best filters. Until such a method exists, the subset of best stack filters and the bounds on this subset derived in this section should suffice. They do lead to a heuristic algorithm in Section 6 which always finds a filter within these bounds.

The subset of best filters that we consider is a significant subset since all its elements are type-3 stack filters and they will, in a sense that will be clear at the end of Section 5, be the best filters which are furthest from the very restrictive class of type-1 and type-2 stack filters. Note that we earlier showed how to find filters which are best type-1 and type-2 filters. An example given at the end of Section 6 will show how the procedure we develop in this and the next section results in a type-3 stack filter which is significantly better than any best type-1 and type-2 filters.

The filters which bound the subset of possible best filters of interest are the filters $S_{f_l}(\cdot)$ and $S_{f_u}(\cdot)$ found in the previous section. The final result of this section will be a proof of the following Existence Theorem, which states that at least one best stack filter exists between these bounds. (The notation $f_b \in \mathcal{F}_1 \cap \mathcal{F}_2$ will be explained later in this section.)

The Existence Theorem for Best Filters:

There exists a best filter $S_{f_b}(\cdot)$ such that $S_{f_l}(\cdot) \leq S_{f_b}(\cdot) \leq S_{f_u}(\cdot)$ and $f_b \in \mathcal{F}_1 \cap \mathcal{F}_2$.

□

In the remainder of this section, we will derive some properties to clarify the behavior of better filters with respect to A, and these properties will then lead to a proof of the Existence Theorem.

Lemma 4.1:

(a) Let $V \subseteq x^N 1 x^N$ and $P\_EXT(one(A)) \subseteq P\_EXT(V)$, and let the stack filter $S_f(\cdot)$ have

$$on(f) = P\_EXT(V) \cup \{D^0(P\_EXT(V)) \cap [x^N 0 x^N - N\_EXT(zero(A))]\}$$

and

$$off(f)=N\_EXT([x^N1x^N-P\_EXT(V)])\cup N\_EXT(zero(A)).$$

Then $S_f(\cdot)$ is the best stack filter among all the stack filters which preserve A and whose *on−sets* in the subspace x^N1x^N are equal to $P\_EXT(V)$.

(b) Let $W\subseteq x^N0x^N$ and $N\_EXT(zero(A))\subseteq N\_EXT(W)$, and let the stack filter $S_f(\cdot)$ have

$$on(f)=P\_EXT([x^N0x^N-N\_EXT(W)])\cup P\_EXT(one(A))$$

and

$$off(f)=N\_EXT(W)\cup\{D^1(N\_EXT(W))\cap[x^N1x^N-P\_EXT(one(A))]\}.$$

Then $S_f(\cdot)$ is the best stack filter among all the stack filters which preserve A and whose *off −sets* in the subspace x^N0x^N are equal to $N\_EXT(W)$.

□

This Lemma can be proven by the same method which we used to prove Lemma 3.2. This results can be visualized with the set diagrams in Figures 4.1 and 4.2, which are similar to those of Figure 3.1. Note that no matter how we expand or shrink the sets $P\_EXT(V)$ and $N\_EXT(V)$, by the root-preservation lemma, $P\_EXT(V)$ must contain $P\_EXT(one(A))$ and $N\_EXT(W)$ must contain $N\_EXT(zero(A))$.

Based on Lemma 4.1, two procedures are now designed to modify a given positive Boolean function f to make it *better*.

Procedure 1 (output: f^*):
Let $V=[on(f)\cap x^N1x^N]$ (input). Then

$$on(f^*)=P\_EXT(V)\cup\{D^0(P\_EXT(V))\cap[x^N0x^N-N\_EXT(zero(A))]\}$$

and

$$off(f^*)=N\_EXT([x^N1x^N-P\_EXT(V)])\cup N\_EXT(zero(A)).$$

□

The filter f^* has an *on−set* which contains more elements of the subspace x^N0x^N than the *on−set* of the original filter f; the *on−sets* of f and f^* contain exactly the same elements of the subspace x^N1x^N. Since this procedure enlarges the on-set of f by extending it "further" into the subspace x^N0x^N, the filter f^* is always better than the filter f. In fact, f^* is a member of the set of

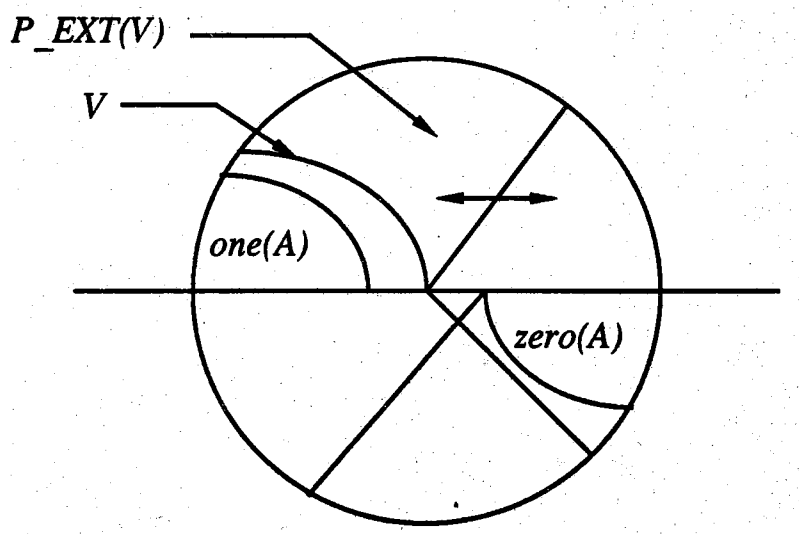


Figure 4.1: This graph is similar to Figure 3.1(a). The set *P_EXT(V)* is used to represent the expansion of the set *P_EXT(one(A))* and the same procedure used to construct f_l is also used to construct a possible better filter with the input *P_EXT(V)*.

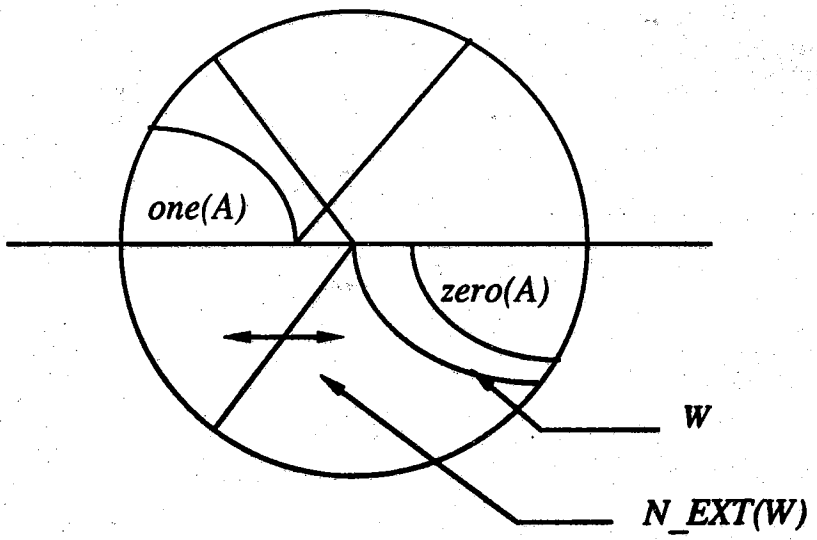


Figure 4.2: This graph is similar to Figure 3.1(b). The set *N_EXT(W)* is used to represent the expansion of the set *N_EXT(zero(A))* and the same procedure used to construct f_u is also used to construct a possible better filter with the input *N_EXT(W)*.

best stack filters whose on-set contains the same elements of x^N1x^N as the on-set of f.

We now state the dual of Procedure 1.

Procedure 2 (output: f^{**}):
Let $W=[off(f)\cap x^N0x^N]$ (input). Then

$$on(f^{**})=P\_EXT([x^N0x^N-N\_EXT(W)])\cup P\_EXT(one(A))$$

and

$$off(f^{**})=N\_EXT(W)\cup\{D^1(N\_EXT(W))\cap[x^N1x^N-P\_EXT(one(A))]\}.$$

□

The filter f^{**} has an *off−set* which contains more elements of the subspace x^N1x^N than the *off−set* of the original filter f; the *off−sets* of f and f^{**} contain exactly the same elements of the subspace x^N0x^N. Since this procedure expanding the off-set of f by extending it "further" into the subspace x^N1x^N, the filter f^{**} is always better than the filter f. In fact, f^{**} is a member of the set of best stack filters whose off-set contains the same elements of x^N0x^N as the off-set of f.

Note that Procedures 1 and 2 always produce filters which as far from or further from the set of type-1 and type-2 filters, respectively, as the filter that is the input to the procedure. In this case, further means that the on-set contains more elements of x^N0x^N, or the the off-set contains more elements of x^N1x^N.

Note that, by the properties of positive Boolean functions, $P\_EXT(V)=V$ in Procedure 1 and $N\_EXT(W)=W$ in Procedure 2.

For simplicity, we define the following notation for the set of all stack filters which preserve the set A and have exactly the same set of elements of x^N1x^N (x^N0x^N) in their on-set (off-set).

$$C_1(V,A)=\{S_f(\cdot)\in ST(A):on(f)\cap x^N1x^N=P\_EXT(V).\}$$

Note that in order to satisfy the conditions $S_f(\cdot)\in ST(A)$ and $on(f)\cap x^N1x^N=P\_EXT(V)$, we must have $V\subseteq x^N1x^N$ and $P\_EXT(one(A))\subseteq P\_EXT(V)$; otherwise, $C_1(V,A)=\varnothing$.

In a similar fashion, define

$$C_2(W,A)=\{S_f(\cdot)\in ST(A): off\,(f)\cap x^N0x^N=N\_EXT(W).\}$$

Note that in order to satisfy the conditions $S_f(\cdot)\in ST(A)$ and $off\,(f)\cap x^N0x^N=N\_EXT(W)$, we must have $W\subseteq x^N0x^N$ and $N\_EXT(zero\,(A))\subseteq N\_EXT(W)$; otherwise, $C_2(W,A)=\varnothing$.

We define the following notation to specify a best filter in $C_1(V,A)$ $(C_2(W,A))$; they are the filters found when Procedure 1 (2) is applied to any filter f in $C_1(V,A)$ $(C_2(W,A))$.

$$\alpha(C_1(V,A))=f^* \text{ and } \beta(C_2(W,A))=f^{**}$$

where

$$on(f^*)=P\_EXT(V)\cup\{D^0(P\_EXT(V))\cap[x^N0x^N-N\_EXT(zero\,(A))]\}$$

$$off\,(f^*)=N\_EXT(x^N1x^N-P\_EXT(V))\cup N\_EXT(zero\,(A))$$

and

$$on(f^{**})=P\_EXT(x^N0x^N-N\_EXT(W))\cup P\_EXT(one\,(A))$$

$$off\,(f^{**})=N\_EXT(W)\cup\{D^1(N\_EXT(W))\cap[x^N1x^N-P\_EXT(one\,(A))]\}.$$

That is, $\alpha(\cdot)$ and $\beta(\cdot)$ are two operators corresponding to the operations of Procedures 1 and 2, respectively.

Note that $S_{f^*}(\cdot)$ and $S_{f^{**}}(\cdot)$ may not be the only *best* stack filters in $C_1(V,A)$ and $C_2(W,A)$, respectively.

We now rewrite Properties 3.1 through 3.4 using the notation just defined.

Lemma 4.2:

(a) $\alpha(C_1(one\,(A),A))=\alpha(C_1(P\_EXT(one\,(A)),A))=f_l$.

(b) $\beta(C_2(zero\,(A),A))=\beta(C_2(N\_EXT(zero\,(A)),A))=f_u$.

(c) $S_{f_{i1}}(\cdot)\in C_1(one\,(A),A)$.

(d) $S_{f_{i2}}(\cdot)\in C_2(zero\,(A),A)$.

□

Recall that f_l and f_u were defined in the previous section. Note that f_l can be constructed by Procedure 1 with input $V=P\_EXT(one(A))$ and f_u can be constructed by Procedure 2 with input $W=N\_EXT(zero(A))$.

Let the sets I_1 and I_2 be two sets in the subspace $x^N 1 x^N$ with $I_1 \subseteq I_2$. Then, the on-set of the filter produced by Procedure 1 with input I_1 is trivially contained in the on-set of the filter produced by Procedure 1 with input I_2. A similar statement applies to Procedure 2. We apply this result in the following Lemma.

Lemma 4.3:

(a) $\forall V \subseteq x^N 1 x^N$ such that $P\_EXT(one(A)) \subseteq P\_EXT(V)$, $f_l \leq \alpha(C_1(V,A))$.

(b) $\forall W \subseteq x^N 0 x^N$ such that $N\_EXT(zero(A)) \subseteq N\_EXT(W)$, $\beta(C_2(W,A)) \leq f_u$.

□

Note that $f \leq g$ if and only if $on(f) \subseteq on(g)$, or if and only if $off(g) \subseteq off(f)$.

This lemma shows that the best filter chosen from each family $C_1(V,A)$ is an upper bound of f_l and the best filter chosen from each family $C_2(W,A)$ is a lower bound of f_u.

In order to specify our concepts more clearly, graphical representations will be used to demonstrate the relation between some specific possible better filters.

Definition 4.1:

Let $(\mathcal{F}, \leq)$ be a partially ordered set and $f_1, f_2, \ldots, f_n \in \mathcal{F}$. If $f_1, f_2, \ldots, f_n$ satisfy the following condition

$$f_1 \leq f_2 \leq \cdots \leq f_n,$$

then we say that $\{f_1, f_2, \ldots, f_n\}$ forms a *chain* with lower bound f_1 and upper bound f_n. The graphical representation of this chain is shown in Figure 4.3.

□

From $C_1(V,A)$ $(C_2(W,A))$, we choose a representative best filter by applying the operator $\alpha(\cdot)$ $(\beta(\cdot))$. Now, we collect a set of best filters by allowing the set $V(W)$ to vary.

Figure 4.3: $\{f_1, f_2, ..., f_n\}$ forms a chain and its chain diagram is shown above.

$$\mathcal{F}_1 = \bigcup_{P\_EXT(one(A)) \subseteq V \subseteq x^N 1 x^N} \{\alpha(C_1(V,A))\}.$$

and

$$\mathcal{F}_2 = \bigcup_{N\_EXT(zero(A)) \subseteq W \subseteq x^N 0 x^N} \{\beta(C_2(W,A))\}.$$

Note that $\mathcal{F}_1$ is the family of all best filters constructed by Procedure 1 with any input $V \subseteq x^N 1 x^N$ such that $P\_EXT(one(A)) \subseteq V$ and $\mathcal{F}_2$ is the family of best filters constructed by Procedure 2 with any input $W \subseteq x^N 0 x^N$ such that $N\_EXT(zero(A)) \subseteq W$.

By Lemma 4.3, we have the following result with this new notation.

Lemma 4.4:

(a) $\forall f \in \mathcal{F}_1$, $f_l \leq f$; that is, f_l is the least element of $\mathcal{F}_1$.

(b) $\forall f \in \mathcal{F}_2$, $f \leq f_u$; that is, f_u is the greatest element of $\mathcal{F}_2$.

□

Graphical representations of $(\mathcal{F}_1, \leq)$ and $(\mathcal{F}_2, \leq)$ are shown in Figures 4.4 and 4.5. In Figure 4.4, f_{t2} is the greatest upper bound of $(\mathcal{F}_1, \leq)$, because $on(f_{t2}) \cap x^N 1 x^N = x^N 1 x^N$ is the largest possible input for Procedure 1. Note that f_u appears as a node in Figure 4.4. Its appearance in $(\mathcal{F}_1, \leq)$ is significant, and the fact that it will always appear in $(\mathcal{F}_1, \leq)$ will be shown in the next lemma.

In Figure 4.5, f_{t1} is the least lower bound of $(\mathcal{F}_2, \leq)$, because $off(f_{t1}) \cap x^N 0 x^N = x^N 0 x^N$ which is the largest possible input for Procedure 2. There is also a significant node in Figure 4.5. It is f_l and its presence will be verified in the next lemma.

Although both $(\mathcal{F}_1, \leq)$ and $(\mathcal{F}_2, \leq)$ completely describe a partial ordering of filters -- in terms of which filters are greater than other filters -- preserving A, they only specify loose bounds on the set of possible best filters. In the remainder of this section, the filters above f_u in Figure 4.4 and the filters below f_l in Figure 4.5 will be truncated, resulting in a set of filters which are invariant under processing by *both* Procedure 1 and Procedure 2. Note that an input Boolean function is called "invariant under the processing of Procedure 1 (Procedure 2)" if the output Boolean function is the same as the input Boolean function.

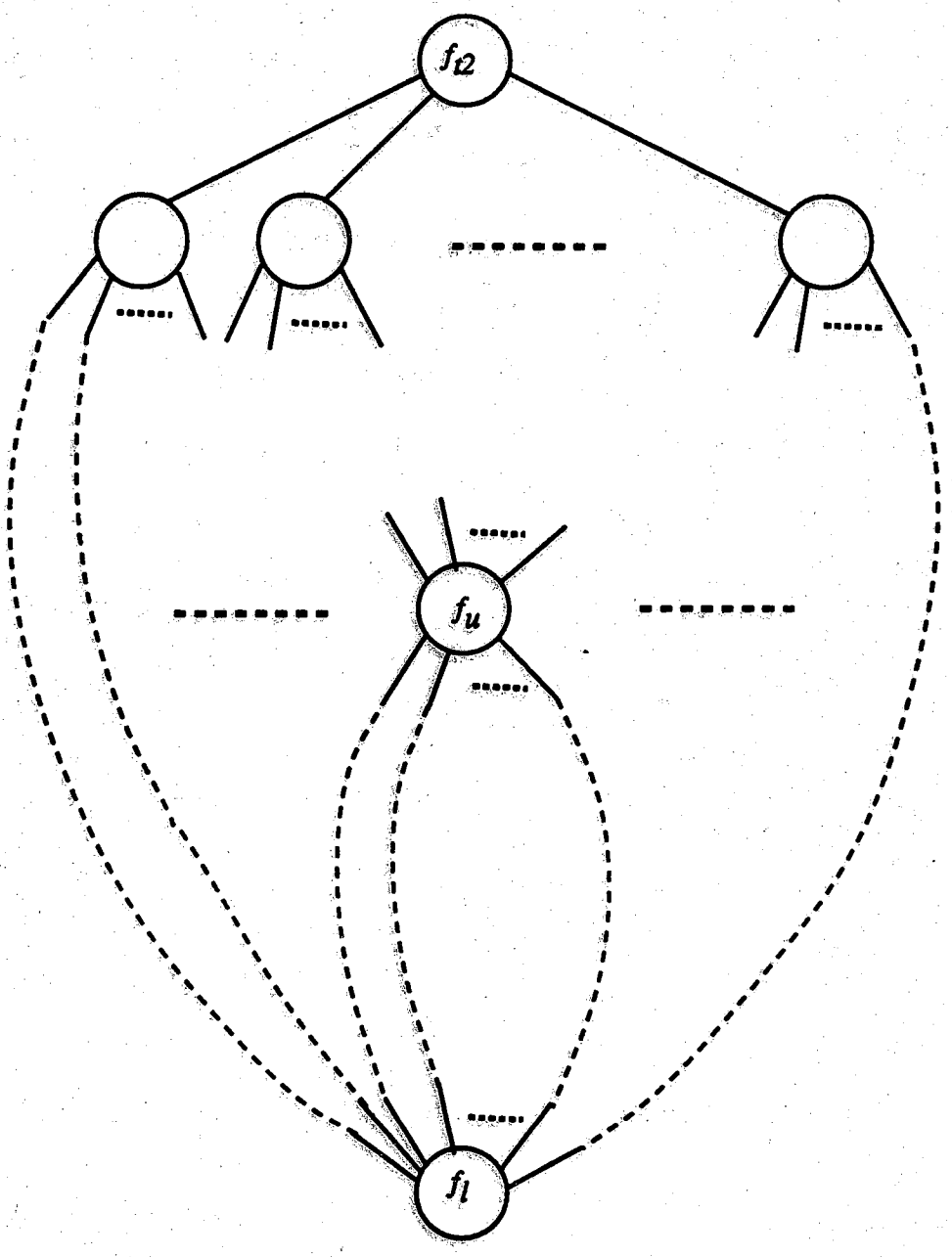


Figure 4.4: The Hasse diagram of $(\mathfrak{F}_1, \trianglelefteq)$

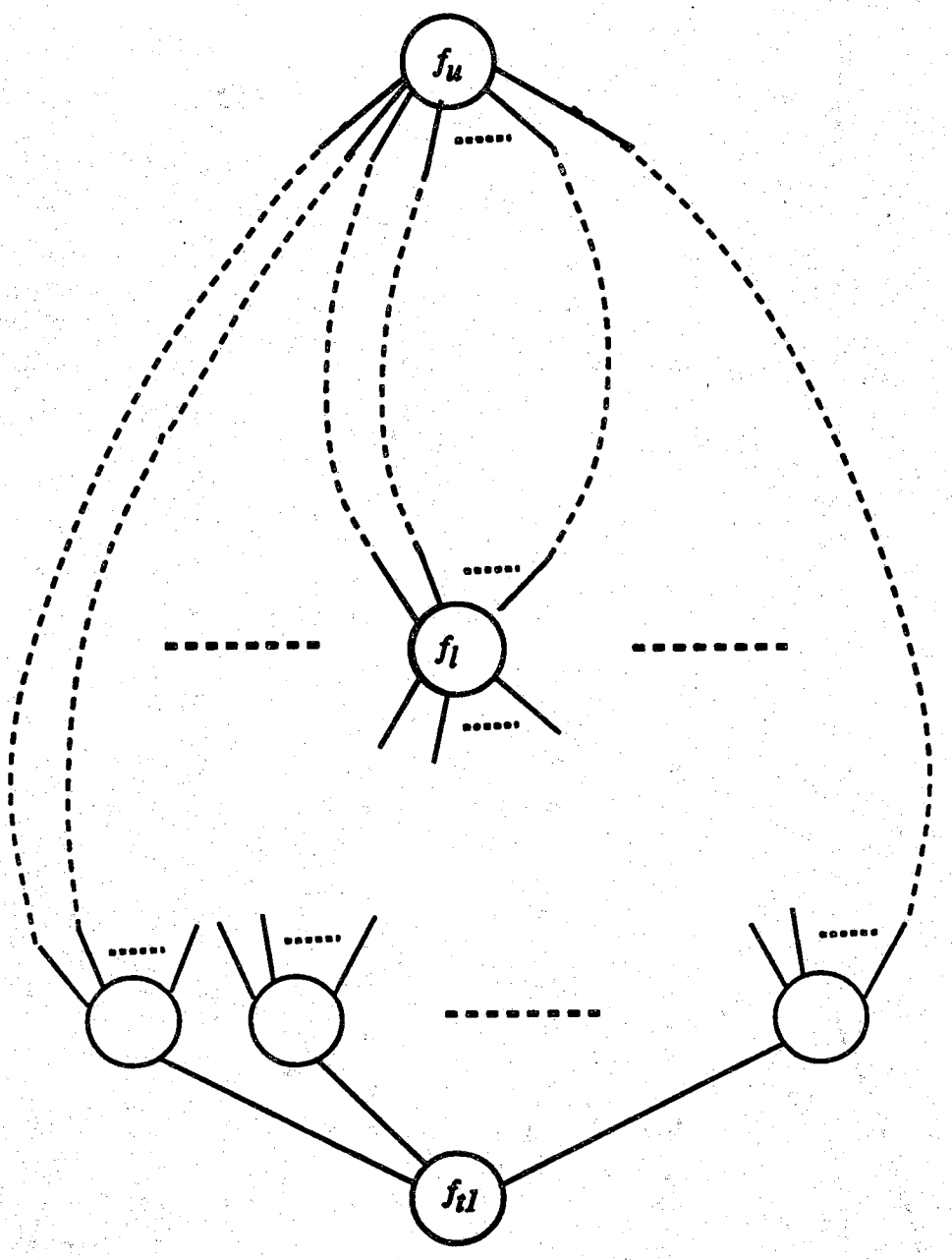


Figure 4.5: The Hasse diagram of $(\mathcal{F}_2, \trianglelefteq)$

Lemma 4.5:

(a) $f_l \in \mathcal{F}_2$.

(b) $f_u \in \mathcal{F}_1$.

Proof:

(a) Let $W = off(f_l) \cap x^N 0 x^N$. We need to show that either $[P\_EXT(x^N 0 x^N - W) \cup P\_EXT(one(A))] \cap x^N 1 x^N = P\_EXT(one(A))$ or $\{N\_EXT(W) \cup \{D^1(N\_EXT(W)) \cap [x^N 1 x^N - P\_EXT(one(A))]\}\} \cap x^N 1 x^N = x^N 1 x^N - P\_EXT(one(A))$. That is, the *on−set* of f_l in the subspace $x^N 1 x^N$ is invariant under the processing of Procedure 2 if the *off−set* of f_l in the subspace $x^N 0 x^N$ is the input to Procedure 2. We pick the first statement to show this invariant case as follows.

$$P\_EXT(x^N 0 x^N - W) \cup P\_EXT(one(A))$$

$$= P\_EXT(on(f_l) \cap x^N 0 x^N) \cup P\_EXT(one(A))$$

$$= P\_EXT(D^0(P\_EXT(one(A))) \cap [x^N 0 x^N - N\_EXT(zero(A))])$$

$$\cup P\_EXT(one(A)) \quad (P\_EXT(\hat{c} \cup \hat{h}) \text{ in Lemma 3.1})$$

$$= \{D^0(P\_EXT(one(A))) \cap [x^N 0 x^N - N\_EXT(zero(A))]\}$$

$$\cup P\_EXT(one(A)) \quad (P\_EXT(\hat{c}) \cup \hat{h} \text{ in Lemma 3.1}).$$

Therefore,

$$[P\_EXT(x^N 0 x^N - W) \cup P\_EXT(one(A))] \cap x^N 1 x^N = P\_EXT(one(A))$$

(b) The dual to the above argument shows that $f_u \in \mathcal{F}_1$.

□

Therefore, f_l and f_u belong to $\mathcal{F}_1 \cap \mathcal{F}_2$; that is, $\mathcal{F}_1 \cap \mathcal{F}_2$ contains at least two elements if $f_l \neq f_u$.

Since every stack filter $S_f(\cdot)$ preserving A belongs to $C_1(on(f) \cap x^N 1 x^N, A)$ and $C_2(off(f) \cap x^N 0 x^N, A)$, and $\alpha(C_1(on(f) \cap x^N 1 x^N, A)) \leq_A f$ and $\beta(C_2(off(f) \cap x^N 0 x^N, A)) \leq_A f$, we have the following lemma.

Lemma 4.6

For all $S_f(\cdot) \in ST(A)$, there exist $f_1 \in \mathcal{F}_1$ and $f_2 \in \mathcal{F}_2$ such that $f_1 \leq_A f$ and $f_2 \leq_A f$.

□

Up to now, we have used Procedures 1 and 2 separately. In the following lemma, the consequences of these procedures together are presented. Alternating applications of Procedure 1 and Procedure 2 will provide a good tool to find a set of better filters which are invariant to both Procedure 1 and Procedure 2.

Lemma 4.7:

(a) $\forall f \in \tilde{\mathcal{F}}_1$, let f be processed by Procedure 2 to obtain g. Then, $g \leq_A f$. That is, $S_g(\cdot)$, $g \in \tilde{\mathcal{F}}_2$, is better than $S_f(\cdot)$.

(b) $\forall f \in \tilde{\mathcal{F}}_2$, let f be processed by Procedure 1 to obtain g. Then, $g \leq_A f$. That is, $S_g(\cdot)$, $g \in \tilde{\mathcal{F}}_1$, is better than $S_f(\cdot)$.

Proof:

(a) Let $V = on(f) \cap x^N 1 x^N$. Since $f \in \tilde{\mathcal{F}}_1$, we have

$$on(f) = P\_EXT(V) \cup \{D^0(P\_EXT(V)) \cap [x^N 0 x^N - N\_EXT(zero(A))]\}$$

and

$$off(f) = N\_EXT([x^N 1 x^N - P\_EXT(V)]) \cup N\_EXT(zero(A)).$$

Let $W = off(f) \cap x^N 0 x^N$, then, after processing by Procedure 2 with input W, we have

$$on(g) = P\_EXT([x^N 0 x^N - N\_EXT(W)]) \cup P\_EXT(one(A))$$

and

$$off(g) = N\_EXT(W) \cup \{D^1(N\_EXT(W)) \cap [x^N 1 x^N - P\_EXT(A)]\}.$$

Note that the off-sets of both f and g contain exactly the same set of elements of $x^N 0 x^N$.

If we can show that $on(g) \cap x^N 1 x^N$ is a subset of $on(f) \cap x^N 1 x^N$, then by Property 2.3, $g \leq_A f$.

$$on(g) = P\_EXT([x^N 0 x^N - N\_EXT(W)]) \cup P\_EXT(one(A))$$

$$= P\_EXT(on(f) \cap x^N 0 x^N) \cup P\_EXT(one(A))$$

$$\subseteq [(on(f) \cap x^N 0 x^N) \cup V] \cup P\_EXT(one(A)).$$

The last statement is true by Lemma 3.1. Now,

$$\{[(on(f)\cap x^N 0x^N)\cup V]\cup P\_EXT(one(A))\}\cap x^N 1x^N$$

$$=V\cup P\_EXT(one(A))$$

$$=V.$$

That is, $on(g)\cap x^N 1x^N \subseteq on(f)\cap x^N 1x^N$.
Therefore, $g\leq_A f$.

(b) The dual of the above argument proves this statement.

□

Lemma 4.7 (a) is based on a procedure in which Procedure 1 is used first and then Procedure 2 is used. Suppose that we begin with a filter h which preserves A. After applying Procedure 1 to h, we obtain f, where, trivially, $f\in\mathcal{F}_1$. If we then apply Procedure 2 to f, we obtain g. These three filters satisfy $g\leq_A f\leq_A h$ and $on(h)\cap x^N 1x^N = on(f)\cap x^N 1x^N \supseteq on(g)\cap x^N 1x^N$. Also, $off(h)\cap x^N 0x^N \supseteq off(f)\cap x^N 0x^N = off(g)\cap x^N 0x^N$.

Similarly, Lemma 4.7 (b) is based on a procedure in which Procedure 2 is used first and then Procedure 1 is used. Suppose that we begin with a filter h which preserves A. After applying Procedure 2 to h, we obtain f, where, trivially, $f\in\mathcal{F}_2$. If we then apply Procedure 1 to f, we obtain g. These three filters satisfy $g\leq_A f\leq_A h$ and $off(h)\cap x^N 0x^N = off(f)\cap x^N 0x^N \supseteq off(g)\cap x^N 0x^N$. Also, $on(h)\cap x^N 1x^N \supseteq on(f)\cap x^N 1x^N = on(g)\cap x^N 1x^N$.

Lemma 4.7 can be generalized as follows:

Lemma 4.8:

(a) $\forall f\in\mathcal{F}_1$, there exists a $g\in\mathcal{F}_2$ such that $g\leq_A f$.

(b) $\forall f\in\mathcal{F}_2$, there exists a $g\in\mathcal{F}_1$ such that $g\leq_A f$.

□

As mentioned before, this result is not good enough to specify a tight bound on the set of possible best filters. In the next lemma, a tighter bound will be obtained.

Recall that "f is invariant under the processing of Procedures 1 and 2" means that if $V=(on(f)\cap x^N 1x^N)$ is the input of Procedure 1, then the output f^* is equal to f and if $W=(off(f)\cap x^N 0x^N)$ is the input of Procedure 2, then the output f^{**} is also equal to f. Also recall that $f\in\mathcal{F}_1\cap\mathcal{F}_2$ if and only if f is invariant under *both* Procedure 1 and Procedure 2.

A graphical representation of $(\mathcal{F}_1 \cap \mathcal{F}_2, \leq)$ is shown in Figure 4.6. The least element and the greatest element of this graph are f_l and f_u, respectively. This graph is much smaller than the graphs of $(\mathcal{F}_1, \leq)$ and $(\mathcal{F}_2, \leq)$. The next Lemma will show that our search for a best filter can be confined to $\mathcal{F}_1 \cap \mathcal{F}_2$.

Lemma 4.9:

(a) $\forall f \in \mathcal{F}_1$, there exists a $g \in \mathcal{F}_1 \cap \mathcal{F}_2$ such that $g \leq_A f$.

(b) $\forall f \in \mathcal{F}_2$, there exists a $g \in \mathcal{F}_1 \cap \mathcal{F}_2$ such that $g \leq_A f$.

Proof:

(a) Let f be processed by Procedure 2 to obtain f_1, and then let f_1 be processed by Procedure 1 to obtain f_2. At the n'th joint application of Procedures 1 and 2, we will have f_{2n} processed by Procedure 2 to obtain f_{2n+1}, and f_{2n+1} processed by Procedure 1 and obtain f_{2n+2}. By Lemma 4.7,

$$\cdots f_{2n+2} \leq_A f_{2n+1} \leq_A f_{2n} \leq_A \cdots \leq_A f_1 \leq_A f_0 = f$$

and

$$(on(f) \cap x^N 1 x^N) \supseteq (on(f_1) \cap x^N 1 x^N) = (on(f_2) \cap x^N 1 x^N) \supseteq \cdots \quad \text{(seq-1)}$$

and

$$(off(f) \cap x^N 0 x^N) = (off(f_1) \cap x^N 0 x^N) \supseteq (off(f_2) \cap x^N 0 x^N) = \cdots \quad \text{(seq-2)}$$

Since seq-1 and seq-2 are decreasing sequences in a bounded space, which is the power set of $\{0,1\}^{2N+1}$ with set inclusion as the partial ordering, there exists a positive integer m such that $\forall n \geq m$

$$(on(f_{2n}) \cap x^N 1 x^N) = (on(f_{2n+1}) \cap x^N 1 x^N) = \cdots$$

and

$$(off(f_{2n}) \cap x^N 0 x^N) = (off(f_{2n+1}) \cap x^N 0 x^N) = \ldots.$$

That is, $f_m \leq_A f$ and f_m is invariant under the processing of Procedures 1 and 2.

Therefore, we have found $f_m \in \mathcal{F}_1 \cap \mathcal{F}_2$ such that $f_m \leq_A f$.

(b) This is the dual of the statement in (a).

□

Proof of the Existence Theorem:

We have been deriving tighter and tighter bounds on a subset of the set of best type-3 filters for preserving a specified set A. We first showed

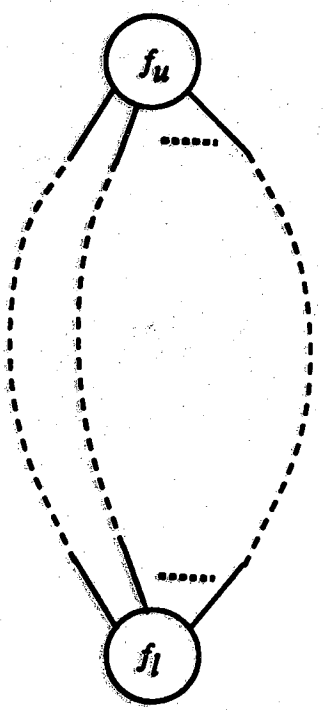


Figure 4.6: The Hasse diagram of $(\mathcal{F}_1 \cap \mathcal{F}_2, \trianglelefteq)$

that we can exclude the type-1 and type-2 filters from consideration as best filters (Lemma 4.2). In Lemma 4.6, we narrow our search for best filters to the set of filters in $\mathcal{F}_1$ or in $\mathcal{F}_2$. The lemma just proven shows that $\mathcal{F}_1 \cap \mathcal{F}_2$ contains a nonempty subset of the set of best filters in $\mathcal{F}_1$ and $\mathcal{F}_2$. We have thus proven the Existence Theorem for Best Filters: the best filters exist in the set $\mathcal{F}_1 \cap \mathcal{F}_2$.

□

Note that since every element in $\mathcal{F}_1 \cap \mathcal{F}_2$ is invariant to both Procedure 1 and Procedure 2, that these filters are as far away from the sets of type-1 and type-2 stack filters as possible. In other words, we can't get any further from type-1 and type-2 by applying Procedures 1 and/or 2 again.

5. The Invariance Theorem

By the Existence Theorem, the set $x^N 0 x^N - N\_EXT(zero(A)) - D^0(P\_EXT(one(A)))$, which is shown in Figure 3.1(a) as area $\hat{f}$, and the set $x^N 1 x^N - P\_EXT(one(A)) - D^1(N\_EXT(zero(A)))$, which is shown in Figure 3.1(b) as area $\hat{c}$, describe bounds on a nonempty set of best filters which are far from the type-1 and type-2 filters. For convenience, we combine Figures 3.1 (a) and (b) in Figure 5.1.

As mentioned in the Existence Theorem, the subset of best filters which are far from type-1 and type-2 filters are invariant under the processing by both Procedure 1 and Procedure 2. In this section, the invariant behavior of the the set of possible best filters lying between f_l and f_u will be exactly specified by a symmetric structure in our set diagrams.

First, let us investigate the behavior between $x^N 0 x^N - N\_EXT(zero(A)) - D^0(P\_EXT(one(A)))$ and $x^N 1 x^N - P\_EXT(one(A)) - D^1(N\_EXT(zero(A)))$.

Lemma 5.1:

$$D^1(x^N 0 x^N - N\_EXT(zero(A)) - D^0(P\_EXT(one(A))))$$

$$= x^N 1 x^N - P\_EXT(one(A)) - D^1(N\_EXT(zero(A)))$$

or

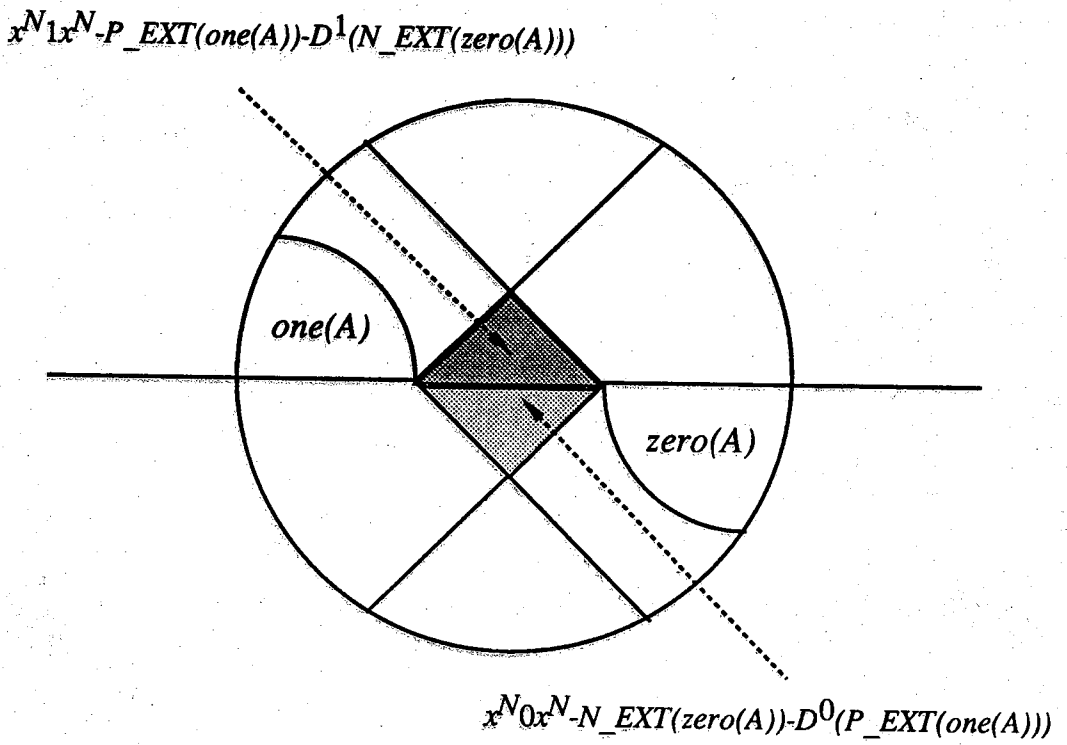


Figure 5.1: Two significant areas are shown in above figure. Those two areas will decide the possible invariant filters under the processing of Procedures 1 and 2.

$$D^0(x^N1x^N\text{–}P\_EXT(one(A))\text{–}D^1(N\_EXT(zero(A))))$$

$$=x^N0x^N\text{–}N\_EXT(zero(A))\text{–}D^0(P\_EXT(one(A))).$$

Proof:

$\forall p1q\in[x^N1x^N\text{–}P\_EXT(one(A))\text{–}D^1(N\_EXT(zero(A)))]$
$\longleftrightarrow$ $\forall r1s\in P\_EXT(one(A));\quad r1s\not\leq p1q,$ and $\forall u1v\in D^1(N\_EXT(zero(A)));\ p1q\not\leq u1v$
$\longleftrightarrow$ $\forall r0s\in D^0(P\_EXT(one(A)));\quad r0s\not\leq p0q,$ and $\forall u0v\in N\_EXT(zero(A));\ p0q\not\leq u0v$
$\longleftrightarrow\forall p0q\in[x^N0x^N\text{–}D^1(P\_EXT(one(A)))\text{–}N\_EXT(zero(A))]$.

Therefore,

$$D^1(x^N0x^N\text{–}N\_EXT(zero(A))\text{–}D^0(P\_EXT(one(A))))$$

$$=x^N1x^N\text{–}P\_EXT(one(A))\text{–}D^1(N\_EXT(zero(A)))$$

or

$$D^0(x^N1x^N\text{–}P\_EXT(one(A))\text{–}D^1(N\_EXT(zero(A))))$$

$$=x^N0x^N\text{–}N\_EXT(zero(A))\text{–}D^0(P\_EXT(one(A))).$$

□

One problem with the two-dimensional diagrams we have been using to illustrate all of the sets of interest to us is that they may lead to the wrong intuition about some of these sets. This is the case, for instance, with the set $x^N0x^N\text{–}N\_EXT(zero(A))\text{–}D^0(P\_EXT(one(A)))$. From the diagram in Figure 5.1 it appears that the negative extension of this set will contain some elements of $D^0(P\_EXT(one(A)))\cap[x^N0x^N\text{–}N\_EXT(zero(A))]$. Similarly it makes it appear that the positive extension of $x^N1x^N\text{–}P\_EXT(one(A))\text{–}D^1(N\_EXT(zero(A)))$ will contain some elements of $D^1(N\_EXT(zero(A)))\cap[x^N1x^N\text{–}P\_EXT(one(A))]$. The following lemma demonstrates that this is not the case.

Lemma 5.2:

(a) For all $p1q\in[x^N1x^N\text{–}P\_EXT(one(A))\text{–}D^1(N\_EXT(zero(A)))]$, $P\_EXT(p1q)\cap\{D^1(N\_EXT(zero(A)))\cap[x^N1x^N\text{–}P\_EXT(one(A))]\}=\emptyset$.

(b) For all $p0q\in[x^N0x^N\text{–}N\_EXT(zero(A))\text{–}D^0(P\_EXT(one(A)))]$, $N\_EXT(p0q)\cap\{D^0(P\_EXT(one(A)))\cap[x^N0x^N\text{–}N\_EXT(zero(A))]\}=\emptyset$.

Proof:

(a) If there exists a $r1s \in P\_EXT(p1q) \cap \{D^1(N\_EXT(zero(A))) \cap [x^N 1 x^N - P\_EXT(one(A))]\}$, then $p1q \leq r1s$ and $r0s \in N\_EXT(zero(A))$. Trivially, $p0q \leq r0s$. Thus, by the Extremal Property, $p0q \in N\_EXT(zero(A))$. That is, $p1q \in D^1(N\_EXT(zero(A)))$. It makes a contradiction with $p1q \in [x^N 1 x^N - P\_EXT(one(A)) - D^1(N\_EXT(zero(A)))]$; i.e., $p1q \notin D^1(N\_EXT(zero(A)))$. Therefore, $P\_EXT(p1q) \cap \{D^1(N\_EXT(zero(A))) \cap [x^N 1 x^N - P\_EXT(one(A))]\} = \varnothing$.

(b) This is the dual of the result in (a).

□

Now, we state the Invariance Theorem as follows.

Invariance Theorem:

Let $f_l \leq f \leq f_u$. Then the following three conditions are equivalent:

(a) $f \in \mathcal{F}_1 \cap \mathcal{F}_2$.

(b) $D^0([on(f) \cap x^N 1 x^N] - [on(f_l) \cap x^N 1 x^N])$
$= [on(f) \cap x^N 0 x^N] - [on(f_l) \cap x^N 0 x^N]$.

(c) $D^1([off(f) \cap x^N 0 x^N] - [off(f_u) \cap x^N 0 x^N])$
$= [off(f) \cap x^N 1 x^N] - [off(f_u) \cap x^N 1 x^N]$.

□

Figure 5.2 is used to specify the different cases arising from the intersection of the on-set and off-set of $f \in \mathcal{F}_1 \cap \mathcal{F}_2$ with the sets $x^N 1 x^N - P\_EXT(one(A)) - D^1(N\_EXT(zero(A)))$ and $x^N 0 x^N - N\_EXT(zero(A)) - D^0(P\_EXT(one(A)))$. We let

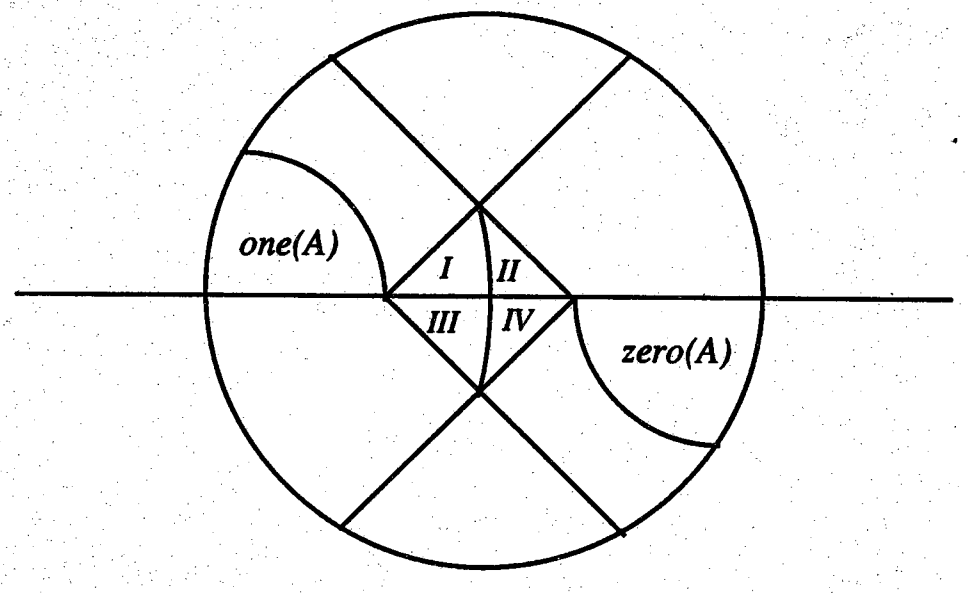
$$I = [on(f) \cap x^N 1 x^N] - [on(f_l) \cap x^N 1 x^N];$$

$$II = [off(f) \cap x^N 1 x^N] - [off(f_u) \cap x^N 1 x^N];$$

$$III = [on(f) \cap x^N 0 x^N] - [on(f_l) \cap x^N 0 x^N];$$

$$IV = [off(f) \cap x^N 0 x^N] - [off(f_u) \cap x^N 0 x^N].$$

The sets I through IV are shown in Figure 5.2. Hence, the Invariance Theorem states that f is invariant under the processing of Procedures 1 and 2 if and



$$I=[on(f)\cap x^N 1 x^N]-[on(f_l)\cap x^N 1 x^N]$$
$$II=[off(f)\cap x^N 1 x^N]-[off(f_u)\cap x^N 1 x^N]$$
$$III=[on(f)\cap x^N 0 x^N]-[on(f_l)\cap x^N 0 x^N]$$
$$IV=[off(f)\cap x^N 0 x^N]-[off(f_u)\cap x^N 0 x^N]$$

Figure 5.2: This graph represents the possible configuration of a positive Boolean function *f* which is invariant under the processing of Procedures 1 and 2.

only if $D^0(I)=III$ and if and only if $D^1(IV)=II$.

Proof of Invariance Theorem:

First we prove that $(b)\Longleftrightarrow(c)$.

Since $(I\cup II)=[x^N1x^N-P\_EXT(one(A))-D^1(N\_EXT(zero(A))]$ and $(III\cup IV)=[x^N0x^N-P\_EXT(one(A))-D^0(P\_EXT(one(A))]$ and by Lemma 5.1 and Disjoint Property, (b) is equivalent to (c).

Second, we prove that $(a)\Longrightarrow(c)$.

Since $f\in\mathcal{F}_1\cap\mathcal{F}_2$,

$\forall p1q\in II$

$\Longleftrightarrow \forall r1s\in P\_EXT(one(A));\ r1s\nleq p1q,\ \forall v1w\in D^1(N\_EXT(zero(A)));$ $p1q\nleq v1w$, and $f(p1q)=0$

$\Longleftrightarrow \forall r0s\in D^0(P\_EXT(one(A)));\ r0s\nleq p0q,\ \forall v0w\in N\_EXT(zero(A));$ $p0q\nleq v0w$, and $f(p0q)=0$

$\Longleftrightarrow p0q\notin D^0(P\_EXT(one(A)))$, $p0q\notin N\_EXT(zero(A))$, and $f(p0q)=0$

$\Longleftrightarrow \forall p0q\in IV$.

Note that "$f(p0q)=0\Longrightarrow f(p1q)=0$" needs to be specified in detail in the previous statements. We know that this special case can not be shown by using stacking property [1-3]. Recall that $f\in\mathcal{F}_1\cap\mathcal{F}_2$. Trivially, $f\in\mathcal{F}_2$. Since we know that $p0q\in(off(f)\cap x^N0x^N)$ and $p1q\in(I\cup II)$. Thus, by the processing of Procedure 2, $p1q\in[off(f)\cap(I\cup II)]$. Therefore, $f(p1q)=0$.

Finally, we show $(c)\Longrightarrow(a)$.

Since we have shown $(b)\Longleftrightarrow(c)$, we can combine the facts (b) and (c) at same time to show that (a) is also true. The method of the proof is same as the method used in the proof of Lemma 4.5.

Let $V=(on(f)\cap x^N1x^N)$ and $W=(off(f)\cap x^N0x^N)$. We need to show that $x^N0x^N\cap on(f^*)=x^N0x^N\cap on(f)$ and $x^N1x^N\cap off(f^{**})=x^N1x^N\cap off(f)$ where f^* is obtained from Procedure 1 with input V and f^{**} is obtained from Procedure 2 with input W; that is, f is invariant under the processing of Procedures 1 and 2 or $f\in\mathcal{F}_1\cap\mathcal{F}_2$.

Since $P\_EXT(V)$ and $N\_EXT(W)$ entirely lie in subspaces x^N1x^N and x^N0x^N, we can simply show that $x^N0x^N\cap on(f)=D^0(P\_EXT(V))\cap[x^N0x^N-N\_EXT(zero(A))]$ and $x^N1x^N\cap off(f)=D^1(N\_EXT(W))\cap[x^N1x^N-P\_EXT(one(A))]$.

$$D^0(P\_EXT(V))\cap[x^N0x^N-N\_EXT(zero(A))]$$

$$=D^0(P\_EXT(one(A))\cup I)\cap[x^N0x^N-N\_EXT(zero(A))]$$

$$=[D^0(P\_EXT(one(A)))\cup D^0(I)]\cap[x^N0x^N-N\_EXT(zero(A))]$$

$$=[D^0(P\_EXT(one(A)))\cup III]\cap[x^N0x^N-N\_EXT(zero(A))]$$

$$=\{D^0(P\_EXT(one(A)))\cap[x^N0x^N-N\_EXT(zero(A))]\}$$

$$\cup\{III\cap[x^N0x^N-N\_EXT(zero(A))]\}$$

$$=[on(f_l)\cap x^N0x^N]\cup III$$

$$=on(f)\cap x^N0x^N.$$

Similarly, $D^1(N\_EXT(W))\cap[x^N1x^N-P\_EXT(one(A))]= x^N1x^N\cap off(f)$. Therefore, the Invariance Theorem has been proven.

□

Note that $I\cup II = x^N1x^N-P\_EXT(one(A))-D^1(N\_EXT(zero(A)))$ and $III\cup IV = x^N0x^N-N\_EXT(zero(A))-D^0(P\_EXT(one(A)))$.

Example 5.1:

Using the same data as in Example 3.1, we find that

$$I\cup II=\{01101,01110,10101,10110\}$$

and, trivially,

$$III\cup IV=D^0(I\cup II)=\{01001,01010,10001,10010\}.$$

By the Invariance Theorem, we can easily determine the structure of $\mathcal{F}_1\cap\mathcal{F}_2$. There are 16 elements in $\mathcal{F}_1\cap\mathcal{F}_2$. If we don't choose any elements from $I\cup II$, we get f_l; choose any one element, say v, and we get four possible invariant filters constructed by the following scheme:

$$on(f)=on(f_l)\cup\{v\}\cup D^0(\{v\});$$

..., pick four elements (all of the four) and we get f_u. Thus, the cardinality of $\mathcal{F}_1\cap\mathcal{F}_2$ is

$$\binom{4}{0}+\binom{4}{1}+\binom{4}{2}+\binom{4}{3}+\binom{4}{4}=16.$$

Note that the four elements in $I\cup II$ are incomparable.

□

The Invariance Theorem reveals a significant phenomenon. To increase or to decrease the portion of the on-set (off-set) of an invariant positive Boolean function in the set $x^N1x^N-P\_EXT(one(A))-D^1(N\_EXT(zero(A)))$ requires simultaneously increasing or decreasing the portion of the on-set (off-set) of that positive Boolean function in the set $x^N0x^N-N\_EXT(zero(A))-D^0(P\_EXT(one(A)))$. Note that the cardinality of set *I* is equal to the cardinality of *III* and the cardinality of *II* is equal to the cardinality of *IV*. In the next section, we will use this concept with Property 2.3 to propose a heuristic method which will allow us to construct a near-best filter, which means it will have very few, but possibly not the minimum number, of false memories.

We now state some special cases of the above theorem. These cases are interesting since they lead to only one or two filters as the candidate best filters.

Corollary 5.1:

The following conditions are equivalent:

(a) $x^N0x^N-N\_EXT(zero(A))-D^0(P\_EXT(one(A)))=\emptyset$.

(b) $x^N1x^N-P\_EXT(one(A))-D^1(N\_EXT(zero(A)))=\emptyset$.

(c) $\mathcal{F}_1\cap\mathcal{F}_2=\{f_l\}=\{f_u\}$.

□

Thus, if region *III*∪*IV* is empty, or if *I*∪*II* is empty, then $f_l = f_u$ and we have found a desired best filter which is furthest from the set of type-1 and type-2 filters.

The following corollary is another stating Corollary 5.1.

Corollary 5.2:

The following conditions are equivalent:

(a) $D^0(x^N1x^N-P\_EXT(one(A)))\subseteq N\_EXT(zero(A))$.

(b) $D^1(x^N0x^N-N\_EXT(zero(A)))\subseteq P\_EXT(one(A))$.

(c) $f_l=f_u$.

□

If there is only one element in either *I*∪*II* or *III*∪*IV*, f_l and f_u are the only possible best filters which are as far as possible from the set of type-1 and type-2 filters. This is stated more precisely as follows.

Corollary 5.3:

The following conditions are equivalent:

(a) $|x^N 1 x^N - P\_EXT(one(A)) - D^1(N\_EXT(zero(A)))| = 1.$

(b) $|x^N 0 x^N - N\_EXT(zero(A)) - D^0(P\_EXT(one(A)))| = 1.$

(c) $|\mathcal{F}_1 \cap \mathcal{F}_2| = 2.$

□

6. The Heuristic Algorithm

By the Existence and Invariance Theorems, we know that we can find a best filter between f_l and f_u and that this filter satisfies the invariant property. We, as yet, do not have an algorithm which is guaranteed to find one of these best filters.

In this section, we will specify an algorithm which improves upon f_l or f_u to find better filters. The algorithm does not necessarily terminate, though, with a best filter.

The algorithm starts with either f_l or f_u and then expands its on-set or off-set, respectively. We will consider only the case in which we expand the on-set of f_l; the dual of the algorithm would then begin with f_u.

In the algorithm to expand the on-set of f_l, we need only consider adding elements from $I \cup II$ to the on-set. Note, though, that adding elements from $I \cup II$ also entails adding the dual element in $III \cup IV$. These two statements are consequences of the Invariance Theorem.

Definition 6.1:

A vertex v in $I \cup II$ is called an *m−expansion* vertex if $[\,|R(on(f_l) \cup P\_EXT(v))| - |R(on(f_l))|\,] = m.$

□

This definition requires that some method is available for counting the number of roots of f_l and its proposed modifications. At this point, this must essentially be done by brute force if exact results are desired. At least this counting need only be done on a few filters.

For convenience, we sometimes use $R(on(f))$ to represent the root set of $S_f(\cdot)$ instead of $R(f)$.

Example 6.1:

As shown in Example 5.1, $I \cup II = \{01101, 01110, 10101, 10110\}$. We used a computer program to determine the effect of adding each of these elements to the on-set of f_l. The result: **01101**, **10101**, and **10110** are **0**—*expansion* vertices and **01110** is a **185**—*expansion* vertex. Thus, adding **01110** to the on-set of f_l produces a filter which has 185 more roots than f_l. Addition of any of the other vertices has no effect.

□

By Property 2.3, as the portion of $on(f)$ in the subspace $x^N 1 x^N$ decreases, it is possible that the cardinality of the root set also decreases; similarly, and as the portion of $on(f)$ in the subspace $x^N 0 x^N$ increases, the cardinality of the root set may decrease. Ideally, we would like to decrease the portion of $on(f)$ in $I \cup II$ and to increase the portion of $on(f)$ in $III \cup IV$ simultaneously. But, by the Invariance Theorem, we know that this is impossible. That is, both parts increase or decrease simultaneously. In order to find a filter better than f_l, we propose a heuristic algorithm which collects the **0**—*expansion* vertices in $I \cup II$. Addition of these vertices will, by definition, not expand the number of roots, and it may decrease their number as the dual vertices in $III \cup IV$ are added in.

Heuristic Algorithm:

Step-1: Find the set V=the set of all **0**—*expansion* vertices.

Step-2: Let $on(f_h) = on(f_l) \cup P\_EXT(D^0(V))$.

□

An experimental result is shown in Table 6.1. We still use the monotonic signals of length 15; that is, L=15. We also tried a variety of window widths: $2N+1 = 3, 5, 7, 9, 11, 13, 15$. In this desired set of patterns, there are 30 elements. Our goal is to design stack filters which preserve this desired set of patterns and only contain a few false memories. Since we know that there does not exist any closed form to find the cardinality of root set of any stack filter, we use the method of enumeration to find the exact number of root set of each stack filter mentioned in Table 6.1.

Note that each numerical entry in Table 6.1 shows the cardinality of the root set corresponding to a specific window width and a specific stack filter. This experiment shows how well each of the filters we have considered so far performs in terms of the total number of roots preserved. Each filter in the table preserves the desired set of 30 signals corresponding to all possible

Table 6.1

f \ $2N+1$	3	5	7	9	11	13	15
f_{t1}	5842	1046	452	247	156	126	121
f_{t2}	5842	1046	452	247	156	126	121
f_l	1974	369	161	98	76	61	51
f_u	1974	369	161	98	76	61	51
f_h	1974	262	110	72	50	39	30

monotonic signals -- but each one may also have false memories, or extra roots. The smaller the total number of roots, the better the filter. Since $one(A)$ and $zero(A)$ in this example are in some sense symmetric, $|R(f_{t1})| = |R(f_{t2})|$ and $|R(f_l)| = |R(f_u)|$.

Table 6.1 shows how an increase in window width decreases the number of roots, and therefore decreases the number of false memories. Each row of the table is decreasing. Note that f_l and f_u, which are type-3 filters, are significantly better than f_{t1} and f_{t2}, which are type-1 and type-2 respectively. Finally, our heuristic approach yields a filter, f_h, which is better than f_l and f_u.

Note that once the window width has been increased to 15 (the last column of the table), in which case there is full connection between every input and every output (signal length = 15), then f_h has no false memories.

7. Conclusion

In this paper, some properties of stack filter based associative memories were presented. The two key results were called the *Existence Theorem* and the *Invariance Theorem*, since they led to a characterization of a set of best filters which are as far as possible from the sets of type-1 and type-2 stack filters.

We specified that *all* possible best filters which are contained in the family of filters whose *on—sets* contain exactly the same of elements of $x^N 1 x^N$, or whose *off—sets* contain exactly the same elements of $x^N 0 x^N$. The best one of each family is bounded from above by $S_{f_u}(\cdot)$ and from below by $S_{f_l}(\cdot)$, which were defined in Section 3.

With the Existence and Invariance theorems as a guide, a heuristic learning scheme was proposed in Section 6 which is very efficient at eliminating false memories.

Appendix

Proof of Property 3.3:

By construction, $on(f) \cap x^N 1 x^N = P\_EXT(one(A))$ and $N\_EXT(zero(A)) \subseteq off(f)$. Thus, $S_f(\cdot)$ preserves A.

Since $S_f(\cdot)$ is a type-3 stack filter whose type-1 decomposition is

$P\_EXT(one(A)) = on(f_{t1})$, by Property 2.3, $R(f)$ is a subset of $R(f_{t1})$. Therefore, $S_f(\cdot)$ is better than any type-1 stack filter with respect to A. Note that, by Lemma 3.1, we are guaranteed that f is a positive Boolean function.

□

Proof of Property 3.4:

By the same manner used in the proof of Property 3.3, this property can also be shown.

□

References

[1] P.-T. Yu and E.J. Coyle, "The Classification and Associative Memory Capability of Stack Filters," *Proc. of 27th Annual Allerton Conference on Communication, Control, and Computing,* University of Illinois at Urbana-Champaign, Illinois, Sept. 1989; and submitted to *IEEE Transactions on Acoustics, Speech, and Signal Processing.*

[2] P.-T. Yu and E.J. Coyle, "Convergence behavior and N-roots of stack filters," accepted for publication in *IEEE Trans. on Acoustics, Speech, and Signal Processing.*

[3] P.D. Wendt, E.J. Coyle, and N.C. Gallagher, "Stack filters," *IEEE Trans. on Acoustics, Speech, and Signal Processing,* vol. ASSP-34, no. 4, August 1986.

[4] E.J. Coyle, J.-H. Lin, and M. Gabbouj, "Optimal stack filtering and the estimation and structural approaches to image processing," *IEEE Trans. on Acoustics, Speech and Signal Processing,* to appear in Dec. 1989.

[5] N.C. Gallagher and G.L. Wise, "A theoretical analysis of the properties of median filters," *IEEE Trans. on Acoustics, Speech, and Signal Processing,* vol. ASSP-29, no. 6, Dec. 1981.

[6] S.G. Tyan, "Median filtering: deterministic properties," in *Two Dimensional Digital Signal Processing II: Transforms and Median Filters,* T.S. Huang, Ed. New York: springer-Verlag, 1981.

[7] T.A. Nodes and N.C. Gallagher, "Median filters: some modifications and their properties," *IEEE Trans. on Acoustics, Speech, and Signal Processing,* vol. ASSP-30, no. 5, Oct. 1982.

[8] B.L. Montgomery and B.V.K. Vijaya Kumar, "Evaluation of the use of the Hopfield neural network model as a nearest-neighbor algorithm," *Applied Optics,* vol. 25, no. 20, Oct. 1986.

[9] J.P. Fitch, E.J. Coyle, and N.C. Gallagher, "Median filtering by threshold decomposition," *IEEE Trans. on Acoustics, Speech, and Signal Processing,* vol. ASSP-32, no. 6, Dec. 1984.

[10] J.P. Fitch, E.J. Coyle, and N.C. Gallagher, "Threshold decomposition of multidimensional rank order operations," *IEEE Transactions on Circuits and Systems,* vol. CAS-32, no. 5, May 1985.

[11] S. Muroga, *Threshold Logic and its Applications,* New York: Wiley, 1971.

[12] M. Gabbouj and E.J. Coyle, "Minimum mean absolute error stack filtering with structural constraints and goals," *IEEE Trans. on Acoustics, Speech and Signal Processing,* to appear in June 1990; and "Design of optimal stack filters with structural constraints under the MAE criterion," *Proc. of the 32'nd Midwest Symposium on Circuits and Systems,* Univ. of Illinois, Urbana, IL, Aug. 14-16, 1989.

[13] G.R. Arce and N.C. Gallagher, "State Description for the root-signal set of median filters," *IEEE Trans. on Acoustics, Speech, and Signal Processing,* vol. ASSP-30, no. 6, Dec. 1982.

[14] J.P. Fitch, E.J. Coyle, and N.C. Gallagher, "Root properties and convergence rates of median filters," *IEEE Trans. on Acoustics, Speech, and Signal Processing,* vol. ASSP-33, no. 1, Feb. 1985.

[15] Z. Agur, A.S. Fraenkel, and S.T. Klein, "The number of fixed points of the majority rule," *Discrete Mathematics* 70 (1988) 395-402.

[16] D.H. Yom and S. Ann, "Directed Graph Representation for the Root Signal Set of Median Filters," *IEEE Proceedings,* vol. 75, no. 11, pp. 1542-1544, Nov. 1987.